

## Expressing Parallel Programs Using Geometric Representation: Case Studies

Brian J. d'Auriol\*

Department of Mathematics and Computer Science  
The University of Akron  
Akron, Ohio, 44325, USA  
bdauriol@cs.uakron.edu

### Abstract

A new method for the specification of parallel programs is presented by example case studies. The essential focus of software development is the expression of groupings of program fragments and the inter-, intra-relationships thereof. Groups of computations are mapped to particular integer points contained in one or more polytopes. Spatial, temporal, hierarchal or other types of relationships over these polytopes can be represented in this framework. An informal review of the geometrical semantics for such geometrical representations is provided in this paper. Practical observations are presented of, in particular, how parallel programs can be expressed in this framework.

**Key Words:** Parallel programming, Geometric representation.

### 1 Introduction

Despite the many advances in compiling technologies and parallel programming language design, parallel programming is still widely regarded as difficult, time consuming and economically costly, particularly when compared to non-parallel program development. Of interest are techniques which may reduce the cost of parallel program design.

In [1], a new *geometric* framework for parallel program representation is proposed to address the difficulties of parallel programming. The essential focus of this work is the expression of collections of computations and the inter-, intra-relationships thereof. Collections of computations are encapsulated by geometric objects, termed *polytopes*, wherein particular computations are mapped to selected integer points inside of the polytopes. Spatial relationships can be quantified by the dimension of the geometric object and by the spatial orientation of the contained computations. Temporal, hierarchal or other relationships can also be represented by superimposing dependency graphs over the geometric object(s). The notion of such collections of program fragments is central to human perception of program design, for example, the separation of program

fragments into modules, objects, tasks, packages, functions, procedures and subroutines, etc. The software development process is thus completed by (a) specifying one or more groupings by describing polytopes and (b) constructing relationships between particular computations and between polytopes. A complete program may be represented by hierarchies of polytopes.

Geometric representation of programs has been established by advances in parallelizing compiler research, in particular, the Polytope Model [2, 3, 4]. In the Polytope Model, programs, expressed in some language are firstly translated to a geometric representation. Optimization may be performed in the geometric domain and lastly, a translation back to the language representation completes the process. The typical benefit of these geometric representations is the precise analysis of the parallelism and inhibiting factors (i.e., dependencies) that can be performed in the geometric domain. The work in [1] extends the Polytope Model by presenting the semantics of geometric representation of program with the focus that the *human initially expresses the solution to a given problem geometrically*.

This paper presents example-based case studies of how initial geometric formulation of problem solutions can be done and what linguistic representations may be derived from such geometric representations. Since the proposed geometrical model is recent and may be regarded as in preliminary stages of development, the presented case studies focus on basic representation issues. The utility for application to parallel program development is implied by these cases studies.

This paper is organized as follows. A review of geometric semantics given in [1] is presented in Section 2. Case studies are presented in Section 3 and conclusions are given in Section 4.

### 2 Geometric Semantics Review

Geometric Semantics is considered in two cases: (a) the geometric specification has an induced semantics due to an initial linguistic formulation of the program and its subsequent translation from the linguistic to the geometric domain, or (b) the initial program specification is given in the geometric domain. Geometric semantics

\*Partially supported by The University of Akron, Grant FRG 1391

due to the first case is termed *linguistic carried semantics*. In the second case, the initial program specification may semantically conform to a linguistic carried semantics. However, it is also possible that the initial program specification may not so conform. Geometric semantics of the latter are referred to as: *non-linguistic carried semantics*.

The following loop nest is considered for the linguistic carried case.

```

for I1 = fl1(l1) to fu1(u1) step s1
  for I2 = fl2(i1, l2) to fu2(i1, u2) step s2
    ...
    for Ip = flp(i1, i2, ..., ip-1, lp) to fup(i1, i2, ..., ip-1, up) step sp
       $\hat{S} = (s_1, s_2, \dots)$ 

```

where each  $I_j, j \in \mathbf{Z}, 1 \leq j \leq p$ , is a loop-index variable with corresponding integer  $i_j$  values given by  $f_{l_j}, f_{u_j}$  and  $s_j, s_j \neq 0$ ,  $f$  denotes a parameterized affine integer function and  $\hat{S}$  is an ordered list of individual program statements,  $S_k$ . In this paper, it is assumed that  $\hat{S}$  is a totally ordered set consisting of expression evaluations or another loop-nest. These restrictions simplify the presentation in this paper. This loop model is denoted by  $\mathcal{L}$ .

In [1], details regarding the geometric object that represents  $\mathcal{L}$  are given. Essentially, instances of  $\hat{S}$  are mapped to specific integer points in an established polytope and the inclusion of temporal relationships over the polytope provides for an *active polytope* representation. Hence, active polytopes are geometric objects which have an orientation determined from  $\mathcal{L}$  and moreover have contained temporal relationships, in the form of dependency graph(s), also induced by  $\mathcal{L}$ . Intuitively, the properties of such an active polytope are fully determined from the corresponding linguistic expression of the program.

### 3 Case Studies

#### 3.1 Shellsort

This case study presents Shellsort considered as groupings of geometric objects. In this example, let the increment sequence be (5,3,1). The usual insertion sort is employed for all sub-lists.

Figure 1 shows the geometric representation of the objects implied by the increment sequence. In (a), there are five individual computations, each of which is an insertion sort. The five computations have no data dependencies, consequently, no temporal ordering between these computations is required. Similarly, in (b), there are three independent computations while in (c), there is a single computation. The numbering system indicates the coordinates of the various points to which computations have been mapped to and is arbitrary in this example.

Although the computations may proceed independently for each distinct value in the increment sequence, the computations between the different values are temporally related. Let a polytope in  $\mathbf{R}^1$  represent the five computations in Figure 1(a), and let two other distinct polytopes in  $\mathbf{R}^1$  represent the computations illustrated in (b) and (c) respectively. The three polytopes are shown in (d) as thick black lines (and as a single point). The temporal relationships between these polytopes are shown by a dashed arc in (d).

In insertion sort, the first computation of an iteration is a comparison and a shift conditional on that comparison. For subsequent iterations where the sorted sub-list consists of  $n$  items, there is at most  $n$  computations and at best 1 computation.

Geometrically, let the state of each of the sub-lists represent an index into one dimension (the independent dimension) and the computations (collectively including comparisons and shifts) represent an index into a second dimension (the dependent dimension). The spatial relationships described above result in a triangle in two dimensional space, as shown in Figure 2 for a sort requiring only six iterations.

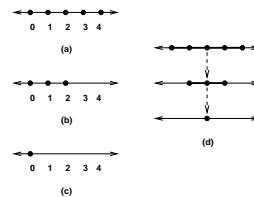


Figure 1: Shell sort increment sequence geometric representation.

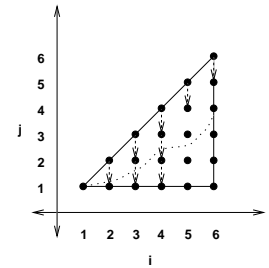


Figure 2: Insertion sort geometric representation.

There are temporal relationships existing within a single iteration. The best case corresponds to those computations mapped to the hypotenuse of the triangle in Figure 2 whereas the computations mapped to the bottom (i.e., for  $j = 1$ ) of the triangle reflect the worst case. Clearly, the temporal ordering is such that the  $j$ th computation must precede the  $j - 1$ th computation,  $1 \leq j \leq i$  for the  $i$  iteration. These temporal relationships are shown in Figure 2 by dashed arcs in the negative  $j$  direction (only a representative subset of such arcs are illustrated). The dotted line in Figure 2 illustrates a hyper-plane cut that, combined with the temporal relationships, represents a particular set of required computations.

There is also a temporal relationship between iterations. Let there be  $n$  one dimensional polytopes such that all computations for each iteration are contained in a distinct polytope. Clearly, the triangle representing the full spatial region of computation will be the convex hull of these  $n$  one dimensional polytopes. Figure 3

shows the six one dimensional polytopes, the temporal relationships between them and the bounding triangle as the convex hull. It is important to note that although the polytopes are of one dimension, they do exist in two dimensional space.

Figure 4 illustrates the combined hierarchal geometric representation for Shellsort.

A common coding of the insertion sort appears in Figure 5 [5]. Much of the previous discussion relating to the geometric representation of Shellsort can be easily related to the code in this figure. For example, the doubly nested `for`-loop with the given constraints upon `i` and `j`, but not including the conditional comparison of the array keys, represent the triangle of Figure 2 together with all temporal relationships as previously discussed and as shown in Figures 2 and 3. The conditional comparison of the array keys represents the hyper-plane cut (the dotted line in Figure 2). The combination of this conditional comparison together with the swap operation forms a single computation that is mapped to each point in the geometric domain.

In summary, there is a single active polytope corresponding to each value in the increment sequence. Each such active polytope has a sequence of instructions (insertion sort) mapped to each contained point in that polytope. Each instruction itself represents a distinct active polytope that has a two-instruction sequence (comparison and conditionally a swap) mapped to each point in that polytope. The step size is 1 in all cases and particular temporal relationships are included. Thus, Shell sort exists in  $\mathbf{R}^4$  with two dimensions for insertion sort, one dimension for each value in the increment sequence and one dimension for the increment sequence itself.

This example provides insight into how one may construct an initial specification of the Shellsort algorithm in the geometric domain. In particular and most importantly, no language representation of the algorithm was considered during the construction of the geometric specification. The algorithm was subdivided into hierarchal geometric objects and relationships within and between these geometric objects were defined. Only the groupings of the computations were studied. These groupings can be described by poly-

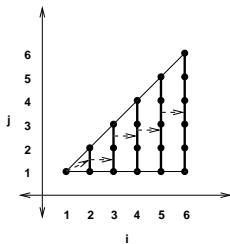


Figure 3: Temporal relationships between iteration polytopes.

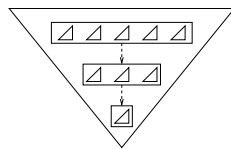


Figure 4: The combined hierarchal geometric representation of Shellsort.

```
for (int i=1; i<n; i++)
  for (int j=1; (j>0) && (key(array[j])<key(array[j-1])); j--)
    swap(array[j], array[j-1]);
```

Figure 5: Insertion sort code.

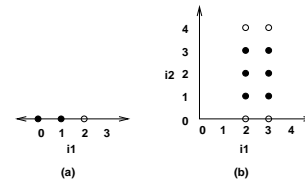


Figure 6: An example of domain specification in different subspaces.

topes with suitable enhancements to represent the various types of possible relationships.

### 3.2 Sub-Space Specification

Figure 6 illustrates an example domain specification in two subspaces. In (a), there is a group of two computations illustrated by the black circles. These computations have been mapped to points (0) and (1) in  $\mathbf{R}^1$ . In (b), six computations, illustrated by the black circles, are grouped together with each computation identified by two indices. These computations exist in  $\mathbf{R}^2$ .

Suppose that these computations reflect a solution to the following problem: choose two (natural) languages and print the first three counting numbers (e.g., in English, one, two, three and in French, *un, deux, trois*). Several query-like questions may be posed. Consider the question: How many languages are represented? and its obvious answer of two. More importantly, the question as well as its answer exist in one dimensional space. Let  $i_1$  in Figure 6 represent the language axis, then, there are two languages grouped together, as represented in (a). A second question may be posed: How many numbers are counted in English? This question and its corresponding answer of three exist in two dimensions, since, the numbers counted is *only meaningful* when also identified by the fact that the counting is done in some language. A last question: How many numbers are counted in any language? also exists in two dimensions for the same reasons. Clearly the index  $i_1$  in Figure 6(a) and (b) must be identically the same. Hence, (a) illustrates a subspace in  $\mathbf{R}^1$  of (b). Note that there is no need to restrict the domain of  $i_1$  to be the same points in each subspace polytope. Figure 6 also shows several white circles. These circles represent non-valid computation points, hence, the polytope presented in (b) contains six valid and four non-valid computation points where special *no-operation* instructions are identified with these non-valid computation points.

A linguistic interpretation is also possible. Consider two `for` loops such that the outer iterates amongst the languages while the inner iterates amongst the num-

bers to be counted. The first question posed above would be reflected at the level of the outer loop only whereas the other two would be reflected at the level of the inner loop.

The polytopes  $P_1$  and  $P_2$  corresponding respectively to Figure 6(a) and (b) are defined as:

$$P_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} i_1 \\ -i_1 \end{bmatrix} \leq \begin{bmatrix} 3 \\ 0 \end{bmatrix} \quad P_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i_1 \\ -i_1 \\ i_2 \\ -i_2 \end{bmatrix} \leq \begin{bmatrix} 3 \\ 2 \\ 4 \\ 0 \end{bmatrix}$$

Given that the *purpose* of such polytope definition reflects descriptive abstractions of the contained computations,  $P_1$  would reflect the abstraction: allow for three natural languages, but implement the solution only for two.  $P_2$  would reflect the abstraction: allow five numbers to be counted, but only implement the solution for three. Furthermore, since  $P_2$  does not describe three consecutive points in the  $i_1$  axis (as does  $P_1$ ), the allowance for five numbers to be counted is only applicable to the two given languages represented by  $i_1 = 2$  or  $i_1 = 3$  in  $\mathbf{R}^2$  subspace.

### 3.3 Specification of Geometric Representation

Two interesting observations can be made regarding the previous case study (Sub-Section 3.2): (a) multiple index domain sets referred to as *bindings* exist and (b) that multiple polytopes may describe the *same* problem solution (although from different viewpoints). For example, Figure 6 showed four types of bindings for  $i_1$ , two for each of the polytopes  $P_1$  and  $P_2$  where, one of these bindings can be described as *static* (e.g., the domain for  $i_1$  in  $P_1$  is statically described by the set  $\{0, 1\}$ ) and the other binding as *dynamic* (e.g.  $P_1$  describes the dynamic binding set  $\{0, 1, 2\}$ ). Furthermore, both  $P_1$  and  $P_2$  describe the same solution. A new mathematical representation given in [1] provides for a general, consistent and integrated means of specifying such types of multiple domain bindings and polytopes in one or more subspaces.

#### 3.3.1 Example of Multiple Domain Specifications

Consider the system:

$$[9, 1, 1] \leq [i_1, i_2] \odot \begin{bmatrix} \{1, 2, 3\} & \{5, 6\} & \{\} \\ \{7, 8\} & \{\} & \{11\} \end{bmatrix} \leq [11, 10, 10]$$

where  $\odot$  denotes a *vector-matrix like* operation which defines plus reduction over the association of static domain sets (specified by the matrix) to the respective indices (specified by the  $I$  vector).

The following six inequalities are specified by this notation:

$$9 \leq i_1\{1, 2, 3\} + i_2\{7, 8\} \leq 11 \quad (1)$$

$$1 \leq i_1\{5, 6\} \leq 10 \quad (2)$$

$$1 \leq i_2\{11\} \leq 10 \quad (3)$$

Inequalities 2 and 3 specify valid and non-valid computation points, i.e. a polytope, in distinctly different subspaces in  $\mathbf{R}^1$ . Note that these inequalities, in addition to subspace specification, also partially specify a region in  $\mathbf{R}^2$ . This aspect is discussed later in the example.

Figure 7(b) and (c) show the graphical representation of the  $\mathbf{R}^1$  subspaces that are determined from Inequalities 2 and 3. Note that the respective polytopes are bounded by 1 and 10. As illustrated in Figure 7(b) by the filled circles ( $i_1 = 5$  and  $i_1 = 6$ ), this polytope abstraction encapsulates the computation group. As before, the set of non-valid computation points are represented by the white circles. However, a degenerate case is exemplified in Figure 7(c) where the polytope abstraction does not sufficiently describe the related set of computations.

A more complicated case is specified by Inequality 1 where valid computation points in  $\mathbf{R}^2$  are specified. Let the set of valid computation points be:  $\{(1, 8), (2, 7), (2, 8), (3, 7), (3, 8)\}$ .

Consider the solution to the system represented by Inequalities 1 through 3, that is, consider the solution given by all inequalities that partially determine a solution in  $\mathbf{R}^2$ . By applying Fourier's Method of Elimination,  $i_1$  and  $i_2$  are determined to be:

$$\begin{aligned} 1 &\leq i_1 \leq 10 \\ \max(1, 9 - i_1) &\leq i_2 \leq \min(10, 11 - i_1) \end{aligned}$$

Note that the domain set for  $i_1$  statically bound in Inequality 1 satisfies Inequality 2 and that the domain set for  $i_2$  statically bound in Inequality 1 satisfies Inequality 3. Consequently, the bindings of Inequality 1 are sufficient to determine a subregion of the solution given by Fourier's Method of Elimination. The solution determined solely by Inequality 1 when the static (local) bindings of  $i_1$  and  $i_2$  are incorporated is:

$$\begin{aligned} 1 &\leq i_1 \leq 3 \\ \max(7, 9 - i_1) &\leq i_2 \leq \min(8, 11 - i_1) \end{aligned}$$

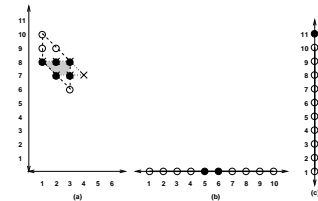


Figure 7: Non-linguistic carried geometric semantics example

```

for i1 = 1 to 3
  for i2 = max(7,9-i1) to min(8,11-i1)
    doit(...)
(a)
for i1 = 1 to 3
  for i2 = 9-i1 to 11-i1
    if (i1,i2) in the valid computation set
      doit(...)
(c)
for i2 = 7 to 8
  for i1 = 9-i2 to 11-i2
    if (i1,i2) in the valid computation set
      doit(...)
(b)
for i1 = 9-i2 to 11-i2 (when i2 bound to {7,8})
  intersection with
for i2 = 9-i1 to 11-i1 (when i1 bound to {1,2,3})
  doit(...)
(d)

```

Figure 8: Linguistic interpretations of Example 4.

This solution region is shown in Figure 7(a) as the shaded area. The filled in circles represent the contained points.

Alternatively, let  $i_1$  be determined from some function of  $i_2$  and also let  $i_2$  be determined from some function of  $i_1$ .

$$9 - i_2\{7, 8\} \leq i_1\{1, 2, 3\} \leq 11 - i_2\{7, 8\} \quad (4)$$

$$9 - i_1\{1, 2, 3\} \leq i_2\{7, 8\} \leq 11 - i_1\{1, 2, 3\} \quad (5)$$

Note that Inequalities 4 and 5 are consistent with computing the solution by Fourier’s Method of Elimination. In fact, each of these inequalities actually specifies the elimination of  $i_1$  and  $i_2$ , respectively.

Note that in Inequality 4,  $i_1$  remains statically bound to the set  $\{1, 2, 3\}$  and in addition, is dynamically bound to the region specified by the following inequalities.

$$\begin{aligned} \text{for } i_2 = 7 \quad & 2 \leq i_1\{1, 2, 3\} \leq 4 \\ \text{for } i_2 = 8 \quad & 1 \leq i_1\{1, 2, 3\} \leq 3 \end{aligned}$$

The polytope resulting from consideration of the dynamic bindings only is shown in Figure 7(a) as the dotted region. This alternative interpretation requires the evaluation of the static bindings (the domain set) for  $i_2$ , but not for  $i_1$ , that is,  $i_1$  may be considered the dependent variable. Similarly, Inequality 5 specifies the following region:

$$\begin{aligned} \text{for } i_1 = 1 \quad & 8 \leq i_2\{7, 8\} \leq 10 \\ \text{for } i_1 = 2 \quad & 7 \leq i_2\{7, 8\} \leq 9 \\ \text{for } i_1 = 3 \quad & 6 \leq i_2\{7, 8\} \leq 8 \end{aligned}$$

This polytope from the consideration of the dynamic bindings is also shown in (a) as the dashed region. The specified polytope from Inequality 1 when considering both type of bindings is the intersection of the dashed and dotted polytopes. The intersection is in fact the same as computed earlier using Fourier’s Method.

Several possible linguistic interpretations of Inequality 1 exists. Four particularly interesting interpretations are subsequently discussed. The first, shown in Figure 8(a), corresponds to the solution region when both static and dynamic bindings are fully taken into account and is obtained by Fourier’s Method of Elimination as described earlier. Recall that from Inequalities 4 and 5, the polytope can also be computed from the intersection of two(convex) polytopes. Figure 8(b) corresponds with Inequality 4 while (c) corresponds with Inequality 5.

When considering the definitions of linguistic carried semantics, the linguistic interpretations in Figure 8(a) though (c) actually specify a dependence of

one variable upon the other that is induced by linguistic carried semantics only. This significance of the non-linguistic semantics is not captured by these linguistic interpretations. The linguistic interpretation shown in Figure 8(d), however, captures the significance of the non-linguistic carried semantics. Note that the specification primitive `intersection with` is an artifact induced by the non-linguistic geometric semantics.

### 3.3.2 A Prototype Software System

This section presents a prototype software system that assists in the exploration of various geometric domain specifications and the corresponding linguistic representations of such specifications<sup>1</sup>. Currently, only the linguistic correspondence in  $\mathbf{R}^p$  is explored and only to the extent of a particular selection of independent and dependent domain sets. The software currently does not support the `intersection with` primitive described earlier nor multiple static bindings. The software does provide for linguistic interpretations of the geometric specification presented earlier for purposes of understandability and linguistic verification of specified computation collections.

In keeping with this purpose, the input is in the form given earlier, however, a small modification to the definition of the matrix has been incorporated. Here, the matrix is a  $p \times p$  matrix of functions such that each column determines a particular single loop nest. The output is a `for` loop nest corresponding to the given input such that successive columns in the matrix specify successive loop nests, respectively. The geometric object (polytope) is thus represented in the output by the bounds of each of the loops, the set of valid computation points are explicitly enumerated in the loop body.

Conceptually, the principal data structure is the `set` which provides for the specification of the domain of the various indices. Collections of `set` are provided by a matrix abstraction. Currently, sets are described by functions that provide for a set’s enumeration. Figure 9 presents the Unified Modeling Language (UML) [6] class diagram for this software.

Consider the re-formulated system corresponding to that used earlier:

$$[9, 9] \leq [i_1, i_2] \odot \begin{bmatrix} \{1, 2, 3\} & \{1, 2, 3\} \\ \{7, 8\} & \{7, 8\} \end{bmatrix} \leq [11, 11]$$

The outer loop,  $i_1$  will be determined from the first column whereas the inner loop,  $i_2$  will be determined from the second column. That is,

$$9 \leq i_1\{1, 2, 3\} + i_2\{7, 8\} \leq 11$$

is explicitly specified twice in this software. Note that here Inequality 1 is used twice as per Inequalities 4

<sup>1</sup>The assistance of S. Saladin and S. Humes who contributed to the program implementation of this prototype is acknowledged.

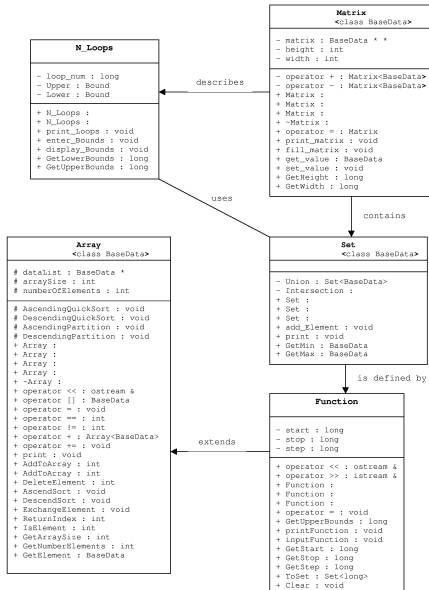


Figure 9: UML class diagram for the prototype software system

and 5. Also note that the set of valid computation points is inherently ‘rectangular’ in  $\mathbf{R}^2$  due to the use of all domain values to specify valid computation points, that is,  $(1, 7)$  is added to the set of valid computation points used above.

Figure 10 shows the output of an execution of the prototype software for this specification. The ‘lower bounds constant matrix’, ‘variable reference matrix’ and ‘upper bounds constant matrix’ correspond exactly to the geometric specification presented above. The complete loop nest is also shown at the bottom of the output.

```

Lower bounds constant matrix:
| 9 9 |

Variable reference matrix:
| f(1,3,1) f(1,3,1) |
| f(7,8,1) f(7,8,1) |

Upper bounds constant matrix:
| 11 11 |

for I1 = 1 to 4
  for I2 = 6 to 10
    if I1 is an element of { 1, 2, 3 }      &&
    if I2 is an element of { 7, 8 }
  
```

Figure 10: Example output

## 4 Summary and Conclusion

This paper proposes, by example, geometric specification as a new method of parallel program specification to address current parallel programming difficulties. The focus of geometrical specification is the construction of spatially, temporally or hierarchal ordered groups of computations. The spatial grouping of com-

putations can be abstracted by enclosing the computations by polytopes. Properties consisting of additional relationships over the enclosed computations may be added to the polytope. Hierarchies of abstractions can be formed by collections of polytopes.

An informal review of linguistic and non-linguistic carried semantics has been provided. Linguistic carried semantics is based on translating an initial linguistic specification of a program segment to a corresponding geometric specification whereas non-linguistic carried semantics is based on the initial specification of groups of program components in the geometric domain.

The primary contributions of this paper are the practical results observed from the three case studies that were presented. In particular, initial program representation in the geometric domain is possible and seems to be a natural extension of human thought expression for certain types of problem solutions. Also, the work reported in this paper supports the idea of ‘query-responsive’ software development, that is, for example, programmers may be able to ‘interact’ with the software by formulating queries. Lastly, mathematical formulations were presented by example as well as a prototype software system that provides for understanding and exploration of this formulation.

## References

- [1] B. J. d’Auriol, S. Saladin, and S. Humes, “Linguistic and non-linguistic semantics in the polytope model,” Tech. Rep. TR99-01, Akron, Ohio, 44325-4002, January 1999.
- [2] C. Lengauer, “Loop parallelization in the polytope model,” *CONCUR ’93*, 1993, E. Best, (ed.), Lecture Notes in Computer Science 715, Springer-Verlag, pp. 398–416, 1993.
- [3] P. Feautrier, “Automatic parallelization in the polytope model,” in *The Data Parallel Programming Model*, G. Perrin and A. Darte, (eds.), Lecture Notes in Computer Science 1132, pp. 79–103, Springer-Verlag, 1996.
- [4] U. Banerjee, *Dependence Analysis*. 101 Philip Drive, Assinippi Park, Norwell, MA, USA, 02061: Kluwer Academic Publishers, 1997.
- [5] C. A. Shaffer, *A Practical Introduction to Data Structures and Algorithm Analysis*. Upper Saddle River, New Jersey, 07458: Prentice Hall, 1997.
- [6] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*. Object Technology Series, One Jacob Way, Reading, MA, 01867: Addison-Wesley, 1999.