Dependence-Free Clustering of Shift-Invariant Data Structures*

Matthias Besch, Hans Werner Pohl

RWCP Massively Parallel Systems GMD Laboratory
Rudower Chaussee 5, 12489 Berlin, Germany

*This work is supported by the *Real World Computing Partnership*, Japan

Abstract: Dependence-free clustering of data structures can be regarded as a general form of alignment and addresses the problem of finding the maximum amount of independent computation on non-connected data sets. The paper presents a unified concept for modelling both data spaces and affine dependence relations with the help of Abelian subgroups of Z^n. This approach allows us to treat alignment at a very high level of abstraction exploiting results of computational algebra.

1 Introduction

Data and code mapping is usually split into several steps which are *alignment*, *distribution* and *assignment*. This paper focuses on a generalisation of alignment, i.e. dependence-free clustering, and does not address aspects of partitioning and assignment. Decoupling alignment has the advantage that load balancing issues and data placement (assignment) optimization need not to be considered. Instead, we can formulate alignment as a *factorization* of a set of data points with respect to given dependences. An equivalence class contains those points that are directly or intermediately connected with each other (Figure 1).

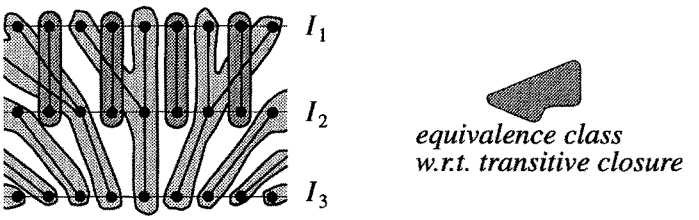

Fig. 1. Alignment as a factorization problem.

Of course, in most practical situations, not all data and dependence structures of some program phase can be aligned without causing a *conflict*. Obviously, a conflict occurs if the alignment results in too few equivalence classes with respect to

the number of envisaged partitions. So, generally, there has to be found a proper subset of data objects and dependence relations.

In our approach both data and dependence structures are represented by sets of integer tuples which we refer to as *index spaces*. So in particular, data domains are not restricted to regular and dense arrays. Translating nested loop programs using affine access functions into this world, data domains and dependence relations show regularities that characterize *shift-invariant structures*.

Since alignment, or dependence-free clustering, only considers the structure of index spaces, the evaluation of boundaries can be postponed to later mapping steps. So, for the remainder of this paper, we restrict our discussion to aligning index spaces in their *infinite extensions*. Therein, our approach is to derive the clustering by means of a *group-theoretic factorization*.

Obviously, we can only give a brief outline here, which we illustrate by way of a single example. For the details concerning the relationship between index spaces and groups, their use for modelling alignment, and various aspects of the implementation using computational algebra, we refer to [2][3][1].

2 Shift-Invariant Data and Dependence Structures

The following simple example (see also Fig. 2) shows a loop nest, whose boundaries are ignored, such that the belonging iteration space is Z^2.

```
forall i, j do
  P[ 2i , 2j ].x = f( L[ 2i+1 , 2j ] , L[ 2i , 2j+1 ] )
  P[ 2i , 2j ].y = g( V[ i-j] )
done
```

The access to the arrays P and L in the first statement is expressed by the *affine* functions

$$(i, j) \rightarrow g(i, j) = (2i, 2j) = (i, j) \cdot \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \text{ for the target operand and}$$

$$(i, j) \rightarrow g^1(i, j) = (2i + 1, 2j) = (i, j) \cdot \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} + (1, 0) \text{ for the first source}$$

argument of f(). Analogously, we define g^2 for the second argument.

Obviously, the image domain of an affine function is always a *coset of a subgroup of* Z^n with respect to componentwise "+" (and vice versa). Therefore, it is quite natural to describe the data spaces and dependence relations by means of group-theoretic notions, i.e. in terms of groups and cosets:

$V = Z$, $P = G1$, $L = G1 + \{(1, 0), (0, 1)\}$, where $G1 = \{(2i, 2j)\}$. Here, $G1$ is a subgroup of Z^2, and L is the union of cosets represented by $(0, 1)$ and $(1, 0)$. The dependence relations can be modelled the same way, such that with $G2 = \{(g, g) | g \in G1\} = \{((2i, 2j), (2i, 2j))\}$ we obtain

$$D_{PL} = G2 + \{up, left\} = G2 + \{((0,0),(1,0)),((0,0),(0,1))\},$$

$$D_{PV} = G_{PV} = \{((2i, 2j), i-j)\}.$$

In general, we consider index spaces I (data-fields and dependences) which are defined by a subgroup $G \subseteq Z^n$ and a set of offsets $S = \{s_1, \ldots, s_k\}$ such that $I = G + S$. Note that I is invariant with respect to all shifts of the group G.

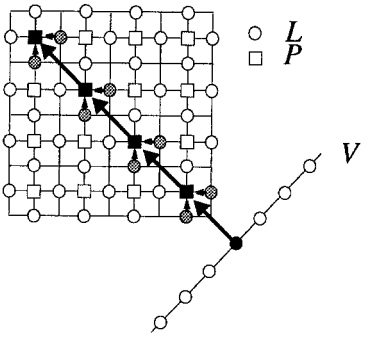

Fig. 2. Index spaces and dependence relations of the example application.

3 Alignment by Group Factorization

The central idea of group-theoretic clustering is elementary. For the lack of space we only consider here the dependence $D_{PV} = G_{PV}$ which is already a group. The problem is to find an appropriate subgroup $A \subseteq G_{PV}$ that serves as a cluster *prototype*. Intuition suggests to take into account all points of the first index space that are dependent on the zero element of the second space, and vice versa. More precisely, we define the groups

$$A_P = \{p|(p,0) \in G_{PV}\} = \{(2i, 2i)\},\ A_V = \{v|((0,0), v) \in G_{PV}\} = \{0\},$$

and construct A as their direct sum $A = A_P \oplus A_V = \{((2i, 2i), 0)\}$. In Fig. 2, A_P and A_V are marked in black.

We can now compute the intended factorization by

$$G_{PV}/A = \{\{((2j + 2i, 2j), i)|j \in Z\}|i \in Z\}$$

which indeed describes the expected alignment classes.

4 Results

In realistic situations, more than one data dependence is to take into account. In addition, a dependence is usually not a group but, for example, a union of cosets. In this case, a number of transformation steps are necessary (see [1][2][3]). First, the selected index spaces and dependences can be naturally decomposed into cosets. Then, an overall space $F = G + S$ is constructed that

extends over all data spaces under consideration. In case S has more than one element the transformation $G + S \Rightarrow (G + \langle S - \{s\}\rangle) + \{s\}$ is computed, with an arbitrary $s \in S$. $\langle S - \{s\}\rangle$ denotes the group generated by $S - \{s\}$. The result is the least coset comprising $G + S$. Generally, we can not assume $G + \langle S - \{s\}\rangle$ to be *transitive*, such that we must compute its transitive closure. The concept of *transitivity of groups* is strongly related to the transitivity of the corresponding (dependence) relations [1]. Since we now have a single coset of a transitive group, $G^* + \{s\}$, we can apply the factorization to G^* following the scheme described in the last section. The result is then shifted back according to the offset s. Considering all dependences of our example, following these transformation steps reults in a facrization that also incorporates the grey points in Fig. 2.

The backbone for an implementation of the alignment process is the unique representation of subgroups of Z^n by means of matrices in so-called *Hermite Normal Form* [4][1][3]. Since in practice, the embedding dimension of the index spaces is rather small (less than ten), there are very efficient operations for the manipulation of groups, such as sum, intersection, projection, test for inclusion, or the computation of cosets. For dealing with the boundaries we use methods of integer linear programming [5].

References

1. M. Besch, H. W. Pohl: *Communication-Driven Alignment of Sparse Data Structures - An Approach Towards Algebraic Mapping*, RWCP Technical Report TR-96014, Japan, http://www.first.gmd.de/promoter/papers/, 1996
2. M. Besch, H. W. Pohl: *Dependence-Free Structured Decomposition of Group-Based Index Spaces*, RWCP Technical Report, Japan, http://www.first.gmd.de/promoter/papers/, 1996
3. M. Besch, H. W. Pohl: *On Using Group-Based Structures for Modelling and Mapping Data Parallel Programs*, Proc. Conf. on Parallel and Distributed Computing and Systems (PDCS'97), Washington, D.C., U.S.A., October 1997
4. H. Cohen: *A Course in Computational Algebraic Number Theory*, Graduate Texts in Mathematics, Springer, 1993
5. P. Feautrier: *Parametric Integer Programming*, Recherche operationelle / Operations Research, vol. 22, no. 3, 1988