Abstract
This paper describes Orca C, the first language built on the Phase Abstractions programming model. The focus is on the syntax and semantics of the data ensembles, a parallel data structuring facility to support machine-independent parallel programs. These features are compared with the data decomposition capabilities of other parallel languages, including Fortran D, Vienna Fortran, and Kali.

1 Introduction
Orca C is the first programming language based on the Phase Abstractions parallel programming model. The importance this model has already been demonstrated: Experiments have shown that it leads to programs that are both efficient and portable. (Scalability is another goal, but this has yet to be demonstrated.) For example, for P processors, the Phase Abstractions version of Livermore's SIMPLE fluid dynamics code has achieved at least P/2 speedup on all types of MIMD parallel machines, including the Sequent Symmetry, the BBN Butterfly GP1000, the Intel iPSC/2 and nCUBE/7 hypercubes, and a mesh machine built from the Transputer T800's [10]; similar results have been reported for the modified Gram-Schmidt method of QR factorization [11]. Though the results do not prove the claims of efficiency and portability, they strongly suggest that a language based on the Phase Abstractions might have these properties. Orca C is such a language.

"Phase Abstractions" is a collective term referring to two types of programming concepts for MIMD parallel computation: the XYZ programming levels and ensembles [14]. The XYZ programming levels structure a parallel program into its constituent parts, including processes, parallel algorithms (phases), data parallel operations and high level control. Ensembles, used to define parallel algorithms (phases), enable the programmer to think of a
computation globally while having local control over the details of general MIMD processes. Three kinds of ensembles are required to define a parallel application: data ensembles define global data structures, code ensembles define the process structure of the computation, and port ensembles specify the communication induced by the data dependencies. Though all three ensemble types are essential for defining parallel algorithms, data ensembles are first among equals, both when designing programs and when understanding Phase Abstractions.

In their most general formulation data ensembles are partitioned data structures [8]. In Orca C, a simple Phase Abstraction language designed for scientific computations, data ensembles are limited to being partitioned arrays. Even so, Orca C’s data ensembles provide greater flexibility than the array partitioning facilities of recent parallel language proposals, e.g., Fortran D. The goal of this paper is to describe the Orca C formulation of data ensembles, present solutions to certain implementation issues, and compare it with related work.

![Figure 1: The XYZ Illustration of SIMPLE.](image-url)

2 Introduction to the Ensemble Abstractions

To describe the role of ensembles in specifying a parallel computation, it is necessary to summarize the details of the XYZ programming levels [13]: A parallel program is viewed as having three kinds of code:

- **process level (X):** composition of instructions to define a lightweight thread,
- **phase level (Y):** composition of processes into a scalable parallel algorithm,
- **problem level (Z):** composition of phases to solve a user’s problem.