Section 1

Overview

This document specifies the form and establishes the interpretation of programs expressed in the High Performance Fortran (HPF) language. It is designed as a set of extensions and modifications to the established International Standard for Fortran (ISO/IEC 1539:1991(E) and ANSI X3.198-1992), informally referred to as "Fortran 90" ([12]). Many sections of this document reference related sections of the Fortran 90 standard to facilitate its incorporation into new standards, should ISO and national standards committees deem that desirable.

1.1 Goals and Scope of High Performance Fortran

The goals of HPF, as defined at an early HPFF meeting, were to define language extensions and feature selection for Fortran supporting:

- Data parallel programming (defined as single threaded, global name space, and loosely synchronous parallel computation);
- Top performance on MIMD and SIMD computers with non-uniform memory access costs (while not impeding performance on other machines); and
- Code tuning for various architectures.

The `FORALL` construct and several new intrinsic functions were designed primarily to meet the first goal, while the data distribution features and some other directives are targeted toward the second goal. Extrinsic procedures allow access to low-level programming in support of the third goal, although performance tuning using the other features is also possible.

A number of subsidiary goals were also established:

- Deviate minimally from other standards, particularly those for FORTRAN 77 and Fortran 90;
- Keep the resulting language simple;
- Define open interfaces to other languages and programming styles;
- Provide input to future standards activities for Fortran and C;
- Encourage input from the high performance computing community through widely distributed language drafts;

- Produce validation criteria;
- Present the final proposals in November 1992 and accept the final draft in January 1993;
- Make compiler availability feasible in the near term with demonstrated performance on an HPF test suite; and
- Leave an evolutionary path for research.

These goals were quite aggressive when they were adopted in March 1992, and led to a number of compromises in the final language. In particular, support for explicit MIMD computation, message-passing, and synchronization was limited due to the difficulty in forming a consensus among the participants. We hope that future efforts will address these important issues.

1.2 Fortran 90 Binding

HPF is an extension of Fortran 90. The array calculation and dynamic storage allocation features of Fortran 90 make it a natural base for HPF. The new HPF language features fall into four categories with respect to Fortran 90:

- New directives;
- New language syntax;
- Library routines; and
- Language restrictions.

The new directives are structured comments that suggest implementation strategies or assert facts about a program to the compiler. They may affect the efficiency of the computation performed, but do not change the value computed by the program. The form of the HPF directives has been chosen so that a future Fortran standard may choose to include these features as full statements in the language by deleting the initial comment header.

A few new language features, including the `FORALL` statement and a few intrinsic functions, are also defined. They were made first-class language constructs rather than comments because they can affect the interpretation of a program, for example by returning a value used in an expression. These are proposed as direct extensions to the Fortran 90 syntax and interpretation.

The HPF library of computational functions defines a standard interface to routines that have proven valuable for high performance computing including additional reduction functions, combining scatter functions, prefix and suffix functions, and sorting functions.

Full support of Fortran sequence and storage association is not compatible with the data distribution features of HPF. Some restrictions on the use of sequence and storage association are defined. These restrictions may in turn require insertion of HPF directives into standard Fortran 90 programs in order to preserve correct semantics.

1.3 New Features in High Performance Fortran

HPF extends Fortran 90 in several areas, including:

- Data distribution features;
- Data parallel execution features;
- Extended intrinsic functions and standard library;
- `EXTRINSIC` procedures;
- Changes in sequence and storage association.

In addition, a subset of HPF suitable for earlier implementation is defined. The following subsections give short overviews of these areas.

In addition to the features that became part of HPF, the HPFF committee considered and rejected many proposals. Suggestions that the committee considered particularly promising for future language efforts to pursue have been collected in a companion document, the HPF Journal of Development [14]. Section 1.7 below gives an overview of this document.

1.3.1 Data Distribution Features

Modern parallel and sequential architectures attain their highest speed when the data accessed exhibits locality of reference. The sequential storage order implied by FORTRAN 77 and Fortran 90 often conflicts with the locality demanded by the architecture. To avoid this, HPF includes features which describe the collocation of data (`ALIGN`) and the partitioning of data among memory regions or abstract processors (`DISTRIBUTE`). Compilers may interpret these annotations to improve storage allocation for data, subject to the constraint that semantically every data object has a single value at any point in the program. In all cases, users should expect the compiler to arrange the computation to minimize communication while retaining parallelism. Section 3 describes the distribution features.

1.3.2 Data Parallel Execution Features

To express parallel computation explicitly, HPF offers a new statement and a new directive. The `FORALL` construct expresses assignments to sections of arrays; it is similar in many ways to the array assignment of Fortran 90, but allows more general sections and computations to be specified. The `INDEPENDENT` directive asserts that the statements in a particular section of code do not exhibit any sequentializing dependences; when properly used, it does not change the semantics of the construct, but may provide more information to the language processor to allow optimizations. Section 4 describes these features.

1.3.3 Extended Intrinsic Functions and Standard Library

Experience with massively parallel machines has identified several basic operations that are very valuable in parallel algorithm design. The Fortran 90 array intrinsics anticipated some of these, but not all. HPF adds several classes of parallel operations to the language definition as intrinsic functions and as standard library functions. In addition, several system inquiry functions useful for controlling parallel execution are provided in HPF. Section 5 describes these functions and subroutines.

1.3.4 Extrinsic Procedures

Because HPF is designed as a high-level, machine-independent language, there are certain operations that are difficult or impossible to express directly. For example, many applications benefit from finely-tuned systolic communications on certain machines; HPF's global address space does not express this well. Extrinsic procedures define an explicit interface to procedures written in other paradigms, such as explicit message-passing subroutine libraries. Section 6 describes this interface. Annex A gives a specific interface for HPF_LOCAL routines and for Fortran 90.

1.3.5 Sequence and Storage Association

A goal of HPF was to maintain compatibility with Fortran 90. Full support of Fortran sequence and storage association, however, is not compatible with the goal of high performance through distribution of data in HPF. Some forms of associating subprogram dummy arguments with actual values make assumptions about the sequence of values in physical memory which may be incompatible with data distribution. Certain forms of **EQUIVALENCE** statements are recognized as requiring a modified storage association paradigm. In both cases, HPF provides a directive to assert that full sequence and storage association for affected variables must be maintained. In the absence of such explicit directives, reliance on the properties of association is not allowed. An optimizing compiler may then choose to distribute any variables across processor memories in order to improve performance. To protect program correctness, a given implementation should provide a mechanism to ensure that all such default optimization decisions are consistent across an entire program. Section 7 describes the restrictions and directives related to storage and sequence association.

1.4 Fortran 90 and Subset HPF

An important goal for HPF is early compiler availability. Because full Fortran 90 compilers may not be available in a timely fashion on all platforms and implementation of some HPF features is more complex than others, we have defined Subset HPF. Users who are most concerned about multi-machine portability may choose to stay within this subset initially. This subset language includes the Fortran 90 array language, dynamic storage allocation, and long names as well as the MIL-STD-1753 features ([27]), which are already commonly used with FORTRAN 77 programs. The subset does not include features of Fortran 90, such as generic functions and free source form, that are not closely related to high performance on parallel machines. Section 8 describes Subset HPF.

1.5 Notation

This document uses the same notation as the Fortran 90 standard. In particular, the same conventions are used for syntax rules. BNF descriptions of language features are given in the style used in the Fortran 90 standard. To distinguish HPF syntax rules from Fortran 90 rules, each HPF rule has an identifying number of the form H*snn*, where *s* is a one-digit major section number and *nn* is a one- or two-digit sequence number. The syntax rules are also collected in Annex **??**. Nonterminals not defined in this document are defined in the Fortran 90 standard. Also note that certain technical terms such as "storage unit" are defined by the Fortran 90 standard; Annex **??** identifies the Fortran 90 rules defining these nonterminals. References in parentheses in the text refer to the Fortran 90 standard.

Rationale. Throughout this document, material explaining the rationale for including features, choosing particular feature definitions, and other decisions is set off in this format. Readers interested in the language definition only may wish to skip these sections, while readers interested in language design may want to read them more carefully. (*End of rationale.*)

Advice to users. Throughout this document, material that is primarily commentary for users (including most examples of syntax and interpretation) is set off in this format. Readers interested in technical material only may wish to skip these sections, while readers wanting a more basic approach may want to read them more carefully. (*End of advice to users.*)

Advice to implementors. Throughout this document, material that is primarily commentary for implementors is set off in this format. Readers interested in the language definition only may wish to skip these sections, while readers interested in compiler implementation may want to read them more carefully. (*End of advice to implementors.*)

1.6 HPF-Conforming and Subset-Conforming

An executable program is HPF-conforming if it uses only those forms and relationships described in this document and if the program has an interpretation according to this document. A program unit is HPF-conforming if it can be included in an executable program in a manner that allows the executable program to be HPF-conforming.

An executable program is Subset-conforming if it uses only the forms and relationships described in this document for Subset HPF (Section 8) and if it has an interpretation under the constraints of Subset HPF. A program unit is Subset-conforming if it can be included in an executable program in a manner that allows the executable program to be Subset-conforming.

(The above definitions were adapted from the Fortran 90 standard.)

1.7 Journal of Development

The HPFF committee considered many proposals, and rejected some that had merit due to external factors (such as lack of agreement in committee). The most promising of these features were collected in the HPF Journal of Development [14]. This section summarizes some of the more detailed proposals.

1.7.1 VIEW Directive

One proposal suggested a directive for relating processor arrangements to each other. This ability is extremely useful in certain applications which use interacting one- and two-dimensional arrays, and has applications for problems consisting of several disjoint data-parallel parts. This feature was carefully discussed, and the committee felt that it was important; however, questions of its implementation complexity eventually caused its rejection.

1.7.2 Nested WHERE Statements

One proposal suggested allowing **WHERE** statements and constructs to be nested within each other. The committee felt that the feature was useful, but declined to include it in HPF because they felt it was too large a change to make to the base language.

1.7.3 EXECUTE-ON-HOME and LOCAL-ACCESS Directives

One proposal suggested a method for specifying the processor(s) to execute a given statement. The same proposal suggested a method for identifying data references which would be mapped to the same processor. In essence, both methods added new directives similar to `INDEPENDENT` (see Section 4.4). Like `INDEPENDENT`, these directives provided information that a compiler might find useful in optimizing the program. Although the committee felt this was an important area to investigate, the proposals were rejected due to technical flaws.

1.7.4 Elemental Reference of Pure Procedures

One proposal suggested allowing elemental invocation of pure procedures (see Section 4.3) under certain conditions. The essential idea was that functions with scalar arguments which could be guaranteed to have no side effects could be invoked elementally, as are intrinsic functions such as `SIN`. The proposal was rejected in a narrow vote, in part because it was seen as too large a change to Fortran 90. After its rejection, the committee voted unanimously to recommend that the ANSI X3J3 committee consider user-defined elemental functions for a future version of Fortran.

1.7.5 Parallel I/O

HPF is primarily designed to obtain high performance on massively parallel computers. Such massively parallel machines also need massively parallel input and output. Accordingly, there were three major proposals to include explicitly parallel I/O features in HPF, as well as several minor variations on the same theme. After much debate, HPFF voted *not* to include I/O extensions in the first version of HPF. Arguments for this position included:

- The diversity of current parallel I/O systems does not suggest any portable abstraction of I/O useful in a language model.
- Fortran I/O is already highly expressive.
- The HPF compiler can optimize the I/O when writing distributed arrays without any extensions to the source language.
- The management of distributed files (and their implementation) is a matter for the operating system, not the language.

Moreover the current lack of extensions does *not* limit features that may be added by system vendors. In particular:

- Vendors are allowed to implement any I/O extensions to the language they may wish. Indeed this would be impossible to prevent. There are simply no special I/O mechanisms mandated by HPF.

- The HPF run-time system may use whatever facilities the operating system provides for accessing "high performance" files, though the HPF language contains no I/O extensions that specifically describe such access.

1.8 Organization of this Document

Section 1, this section, presents an overview of HPF.

Section 2 sets out some basics of HPF, including:

- The reasons for using Fortran 90 as a base language;
- A partial cost model for HPF programs; and
- Lexical rules for HPF directives.

Section 3 describes the facilities for data partitioning in HPF. These include:

- The distribution model;
- Features for distributing array elements among processors;
- Features for aligning array elements which are accessed together; and
- Features for mapping `ALLOCATABLE` arrays, pointers, and dummy procedure arguments.

Section 4 describes the explicitly parallel statement types in HPF. These include:

- The single- and multi-statement forms of the `FORALL` parallel construct;
- Pure functions callable from within `FORALL`; and
- The `INDEPENDENT` assertion for loops.

Section 5 describes new standard functions available in HPF. These include:

- Inquiry intrinsic functions to check system and data partitioning status;
- New computational intrinsic functions and extensions to existing intrinsic functions; and
- A standard library of computational and inquiry functions.

Section 6 describes extrinsic procedures in HPF, particularly the `EXTRINSIC` procedure interface. The material in Annex A builds on this interface.

Section 7 describes the treatment of sequence and storage association in HPF. This includes:

- Limitations on storage association of explicitly distributed variables; and
- Limitations on sequence association of explicitly distributed variables.

Section 8 describes Subset HPF, which may be implemented more quickly than full HPF. This includes:

- A list of Fortran 90 features that are in Subset HPF;
- A list of HPF features that are *not* in Subset HPF; and
- Discussions of why these decisions were made.

Annex A describes a binding for a local execution model for use as an `EXTRINSIC` option. The model implements the Single Program Multiple Data programming paradigm, which has wide (but not universal) applicability.

Annex ?? collects the grammar and syntactic constraints for HPF defined in the main text of this document.

Annex ?? cross-references the BNF terminals and nonterminals defined and used in this document.

The Bibliography provides references to various HPF sources:

- Fortran standards;
- Fortran implementations;
- Books about Fortran 90; and
- Technical papers.