HPFF Meeting Notes for the March 10-12, 1993 Meeting

High Performance Fortran Forum

CRPC-TR93317
May 1993

Center for Research on Parallel Computation
Rice University
6100 South Main Street
CRPC - MS 41
Houston, TX 77005

HPFF Meeting Notes

March 10-12, 1993
Dallas, TX - Bristol Suites Hotel
Notes taken by Chuck Koelbel

Executive Summary

This was the final meeting of the High Performance Fortran Forum, convened to consider public comments to the January draft of the language. Its major result was plans for a new draft of the language specification, to be called “HPF Language Specification, Version 1.0 Final.” The changes were not made to the document directly, due to the number and magnitude of corrections. The new draft should be available May 3, 1993.

Detailed Meeting Notes

Attendees

Alan Adamson (IBM Canada), Robert Babb (Oregon Graduate Institute), Alok Choudhary (Syracuse University), Sia Hassanzadeh (Fujitsu Labs), Tom Haupt (Syracuse University), Don Heller (Shell), Ken Kennedy (Rice University), Bob Knighten (Intel), Chuck Koelbel (Rice University), Ed Krall (MCC), John Levesque (Applied Parallel Research), David Loveman (DEC), Piyush Mehrotra (ICASE), Andy Meltzer (Cray Research), John Merlin (University of Southampton), Ken Miura (Fujitsu America), Jean-Laurent Philippe (Archipel), David Presberg (Cornell University), P. Sadayappan (Ohio State University), Randy Scarborough (IBM), Rob Schreiber (RIACS), Vince Schuster (Portland Group, Inc.), Richard Shapiro (Thinking Machines), Matt Snyder (Lahey Computer), Guy Steele (Thinking Machines), Rick Swift (MasPar), Clemens-August Thole (GMD), Jerry Wagener (Amoco), Joel Williamson (Convex), Min-You Wu (SUNY Buffalo), Mary Zosel (Lawrence Livermore National Lab)

Memorable self-introductions

Don Heller: “I intend to earn my t-shirt.”
Randy Scarborough: “From IBM, who is now offering newly affordable stock prices!”
Joel Williamson: “From Convex, who has even more affordable stock prices.”
Bob Knighten: “From Intel, who is happy to say our stock costs about what IBM used to cost.”

Administrative Issues

The meeting began with the distribution of the HPF t-shirts, featuring a graphic design developed by e-mail. The shirts are gray, with the FORALL synchronization diagram on the front and the slogan “Don’t blame me, I didn’t vote for that feature” on the back. It is expected that these shirts will be appearing at high performance computing conferences for the rest of the year. After a brief discussion of where to go for dinner (Indian food was the choice), the meeting settled down to consider the public comments. Many proposals were made in these comments, and the bulk of the meeting was devoted to ideas from them.

Global Comments, Fortran 90 Issues, and Subset HPF Issues

Mary Zosel presented the comments on HPF in general, Fortran 90 aspects of HPF, and Subset HPF issues. The paragraph numbering corresponds to her overhead slides. All recommendations were from the Fortran 90 and Subset HPF subcommittee, which met the afternoon before.

0. Move the existing appendix (the Journal of Development) to a separate document.

The comment (and several committee members) argued that this section has served its purpose, and shouldn’t be part of the final publication. The subcommittee recommended this proposal by a vote of 4-0. Clemens Thole argued the Journal of Development should be kept, as it was only a small part of the document. Ken Kennedy noted that the issue was whether this should be in the official part of the work, or saved by some other means. Robbie Babb was in favor of keeping the appendix in the journal publication, as long as it was set off from the other parts. A formal vote to separate the documents was taken, passing 14 in favor, 5 opposed, and 4 abstaining.

1. Number syntax rules as in the Fortran 90 standard.

The subcommittee recommended this by a vote of 5-0, “if someone else will do it.” Guy Steele noted that such numbering was easy by modifying the LaTeX macros. He recommended using “H” to identify HPF rules, to disambiguate them from Fortran 90 rules that were also used. The motion was accepted without a formal vote.

2. Create a new appendix, and move rationales from the main chapters to it (as for Appendix C in Fortran 90).

The subcommittee recommended this by a vote of 3-1-1. Piyush Mehrotra wanted the reasoning in-line to avoid flipping pages, but set off from the specification somehow. David Presberg agreed, noting that there were several kinds of asides in the current text.. In response to a comment that nobody read appendices, Vince Schuster responded that implementors know all about Appendix C of the standard, and there are lots of complaints as is.. Andy Meltzer opined that “If you can see what to ignore, it’s OK.” Clemens Thole asked how hard it would be to move the text, and Guy Steele noted it would be relatively easy to create new macros. Randy Scarborough saw a value in teaching our reasons, and suggested publishing a pure reference guide later. A vote was taken to create the new section (or appendix), failing 3-16-4. Rob Schreiber then moved that chapter authors identify material that can be removed, and at their discretion format it in special macros as suggested by Guy. The vote on this proposal was in favor, 15-0-9.

3. Deal with Chapter 7: Eliminate it or move it to an appendix.

The repeated number was explained as "really part of the same issue," since most of the I/O material was in the Journal of Development. Chapter 7 was the I/O chapter, which consisted of three pages explaining why HPF had no I/O extensions. The subcommittee reported two votes: eliminating the material entirely was recommended 4-0-1, while moving it to the appendix was also recommended 5-0-0. Andy Meltzer suggested making it into a rationale section Piyush Mehrotra proposed moving only the real rationale from that chapter to an early section. It was eventually moved and seconded to move the I/O rationale to Chapter 1 ("Overview") , delete Chapter 7 ("I/O"), and keep the I/O material in the Journal of Development. This passed 17-1-6. Chuck Koelbel promised to look into exactly what words should go where.

4. Change "Appendix" to "Annex."

5. Change "Chapter" to "Section."

The two proposals were discussed together. Both represented changes to bring the specification closer to the ISO conventions (and, less fortunately, further from the ANSI conventions). The subcommittee recommended them by votes of 2-0-3, 3-0-2, respectively. The proposals were accepted without a formal vote.

6. Change the default handling for variables to SEQUENCE when no explicit mapping is given for the variable.

6a. Define both SEQUENCE and NOSEQUENCE directives

6b. The SEQUENCE directive means "Variables are sequential except explicitly mapped variables."

The proposals were clearly related. The X3J3 comments on storage and sequence association said HPF had picked the wrong default, which would break old code, and cause a substantial backlash. However, their comments also suggested that "somebody's got to take the plunge away from storage & sequence association." HPFF's stance was to enable aggressive compilers through the language. The subcommittee split on its recommendation, voting 2-2-1. Rob Schreiber noted it would be really bad to change this, because of the implications. Ken Kennedy was sure that no matter what HPF defined, implementors will provide the "-compatible" option. David Loveman argued that HPF should use the Fortran 90 default since it's the base language. Clemens Thole wanted a standard that minimized the new directives, and so favored the current situation. Vince Schuster noted this choice was made to ease portability and bring more applications to HPF. Richard Swift noted that existing codes have association, and therefore HPF compilers need to deal with it. Rob Schreiber noted an asymmetry: if SEQUENCE becomes the default, no compiler will be able to optimize data layout. Jerry Wagener suggested reading the X3J3 comment as "The majority of the committee is in sympathy, but we got beat up badly on this." He went on to suggest that a gentle transition would be better, and 6a is good migration path. His personal comment (although some on the X3J3 committee agreed) was "This needs help, and if you're willing to provide it..." But HPFF would have to do what it thought was right. Mary Zosel characterized the treatment of a variable as a hierarchy of binding

1. Explicitly mapped variables
2. SEQUENCE (or NOSEQUENCE) applied to a name
3. SEQUENCE (or NOSEQUENCE) applied to a scoping unit

4. Compiler switch
5. HPF default

The discussion only set the policy for the outermost binding level.

Finally, a vote was taken to adopt proposal 6a. It passed with 22 in favor, 0 opposed, and 3 abstaining. Another vote was taken to change the default to SEQUENCE, receiving 9 yes votes, 15 no, and 1 abstain. Thus, variables are not sequential by default, but both the SEQUENCE and NOSEQUENCE directives are available to explicitly declare all variables.

Proposal 6b was accepted as an editorial change without formal vote. It was certainly the intended meaning of the directive.

Piyush Mehrotra moved that SEQUENCE/NOSEQUENCE with no name (that is, applying to a whole scoping unit) must be the first line in the scoping unit. His argument was that this avoided having to parse the full program before hitting SEQUENCE. David Presberg argued this was too restrictive, and there are other similar syntactic situations. The vote on Piyush's proposal failed, 2-11-11.

7. Eliminate the HPF Subset.

In view of the fact that early HPF compilers would not implement the full language, this was characterized as a question of "Do we spell subset with a capital or small 'S'?" The subcommittee recommended against eliminating the subset by a vote of 1-4, and the proposal was not moved from the floor. Clemens Thole moved that HPF have a separate document on Subset HPF, but the motion failed for lack of a second. Vince Schuster suggested as an editorial matter that the language specification refer to the sublanguage as "Subset HPF," and this was accepted by acclamation.

10. Add {some form of} MODULE to Subset HPF.

The numbering here drew a number of jokes, including "Someone from Thinking Machines did this" and (from Guy Steele) "Octal habits die hard." The serious reason was that it subsumed proposals 8 and 9 (see below). Long public comments had suggested that Fortran 90 MODULE constructs were very useful and needed in HPF. In particular, there is now no way to do global ALLOCATABLE objects within Subset HPF now. The subcommittee recommended adding MODULE by a vote of 3-2-0. Richard Swift called for a full vote, since MODULE would delay Subset implementations significantly and HPF would be hurt by lack of availability. After some further discussion, the vote to add MODULE was taken, failing by 6-16-3.

9. Add generics to Subset HPF.

Comments from "the math library folks" claimed that they needed them. The subcommittee recommended against the proposal by a vote of 0-2-3. Nobody moved it in full committee, so the proposal died.

8. Add multiple-precision support to Subset HPF

The arguments were similar to the last proposal, and the result was the same. The subcommittee recommended against by a vote of 2-3-0, and it was not moved again in full committee.

11. Position of !HPF$ on a line.

A Cray Research comment had questioned where the !HPF$ directive header could appear on a line. In particular, they challenged the existing constraint that every directive must be a comment line. For example:

```
!HPF$ whatever
```
Should always work

```
stmt1; !HPF$ some directive; stmt2
```
Subcommittee rejected this as inconsistent

```
stmt; !HPF PURE
```
Considered and rejected - see below

```
stmt; !HPF$ ...
```
Considered and rejected - see below

The PURE usage was defeated in subcommittee 0-4. (PURE would be considered later as well.) The last example was defeated 1-3-1, although there was some interest in putting ALIGN and DISTRIBUTE directives on the same line as the variable declarations they applied to. In answer to Rob Schreiber's question "What's the deal?" several committee members noted interest in simple parsing technology. Many projects are doing source-to-source translators for HPF-like directives, and don't need (or want) to handle difficult cases. The last form was moved as a valid HPF syntax. Chuck Koelbel argued this was dangerous, citing the example

```
DO i = 1, n; !HPF INDEPENDENT
  DO j = 1, n
```

(Contrary to what the formatting might imply, INDEPENDENT applies to the "DO j" loop.) The vote on the motion failed with 8 yes votes, 13 no, and 4 abstains.

FORALL, PURE, and INDEPENDENT

After a short coffee break, Chuck Koelbel presented the public comments on the "HPF Programming Model" section and the "Statements" chapter. He began by noting that there were a large number of typographical errors that he would correct without bringing before the committee. One such error that deserved mention was the accidental deletion of the line

```
!HPF$ DISTRIBUTE (CYCLIC) :: c
```

from the programming model examples, leading to several incorrect explanations. At least six reviewers had reported this error. The substantive issues appear below. Recommendations are from the FORALL subcommittee, which had met the previous night.

A Chapter by Any Other Name.

A comment had suggested changing the title of this section ("Statements") to "Concurrent Execution Features." While the subcommittee agreed that the former title was not descriptive, they felt that "concurrent" had the wrong connotations. They proposed the name "Data Parallel Statements and Directives"; the full committee accepted this title by acclamation.

FORALL comment 1. Eliminate FORALL.

This was suggested by employees of Cray Research. The subcommittee responded by voting to retain FORALL by a poll of 5-0-0. The full committee agreed without a formal vote.

FORALL comment 2. Semantics of FORALL are deficient.

This arose in several comments. Most objected that multiple assignments to a single location were not prevented by the FORALL syntax. This was intentional; the restriction against multiple assignment was a semantic one, not a syntactic constraint. This decision had been made at previous HPFF meetings on two grounds: it allowed future extension of FORALL constructs to use private variables, and syntactic constraints were insufficient to guarantee the desired behavior. (There were also syntactic problems in the original proposal, which had effectively disallowed pointer assignment, although it was intended to be included.) Chuck agreed to rewrite the sections to emphasize this constraint more strongly, and include rationale for the decision. No motions were made in committee to resurrect the syntax constraints. Other reviewers had noted discrepancies in the wording of the semantics that were leftovers from the previous constraints. These were fixed as well.

FORALL comment 3. Restrict functions called from the FORALL header to be pure.

John Merlin had noted the current syntactic constraints only said that functions called from the FORALL body had to be pure; there was no constraint on functions called in the FORALL bound and stride expressions. The subcommittee had agreed that the intent was that functions called from the FORALL header should be pure (but no vote was given). Guy Steele objected from the floor, however, that for consistency with DO loops, the constraint should be that the evaluation of any bound or stride expression could not affect any other bound or stride. He agreed that only pure functions should be allowed in the mask expression, as it was conceptually executed for every instance of the FORALL body. After a brief discussion, the full committee sided with Guy, voting 14 yes, 3 no, and 8 abstaining to require pure functions only in the mask expression.

PURE comment 1. Constraints on pure can be simplified.

A comment from J. L. Schonfelder suggested simplifying the constraints on pure functions. Chuck Koelbel and John Merlin were studying his suggestions to determine whether they were in fact equivalent to the current restrictions, and promised to report back. (They later reported that not all the new constraints were statically checkable; in particular the constraint that “a global location cannot be assigned to” was troublesome in the presence of pointers.)

PURE comment 2. Make PURE a full-fledged attribute.

Another comment from Schonfelder was that, since PURE changed the syntax (particularly the constraints) of a function, it should be a full-fledged part of the language. He went further by supplying a suggested syntax, in which PURE was handled in the same way as RECURSIVE. The subcommittee agreed with this recommendation by a vote of 4-0-2. The committee debated the issue, returning to arguments made on this issue in the past. Eventually, a full vote found the committee in favor of PURE as first-class syntax by a vote of 11 yes, 4 no, and 9 abstaining.

PURE comment 3. Replace PURE with ELEMENTAL.

This suggestion, from employees of MasPar, suggested restricting functions called from within FORALL to have scalar arguments only. In addition, they recommended that these functions could be invoked elementally (in the Fortran 90 sense of passing array actual arguments to scalar dummies). They argued that PURE functions with array arguments would not be efficient on certain architectures, due to the necessity of redistribution in some cases. However, elemental functions would be efficiently implementable, and captured most of the intended uses of elemental functions. Others in the subcommittee had argued both points, particularly the second. In the end, the subcommittee was not able to make any recommendation, and asked the full committee for advice. After discussion, a straw vote was taken on whether ELEMENTAL should be further considered in subcommittee, finding 15 in favor, 7 against, and 8 abstaining. A further straw poll showed 9 in favor of adding ELEMENTAL, 4 favoring replacing PURE with ELEMENTAL, and 15 abstaining. The subcommittee met again that evening to reconsider the matter.

INDEPENDENT comment 1. Make INDEPENDENT a full-fledged statement.

INDEPENDENT comment 2. Remove INDEPENDENT.

Not only were these comments related by content, they were made by the same person. Clemens Thole explained Schonfelder's reasoning as follows: If INDEPENDENT were accepted in HPF, then it changed the semantics of the program and thus should be a full statement. However, Schonfelder considered the current INDEPENDENT to be inadequate (in particular, it was not precisely defined); therefore, he felt it better to remove the feature than to struggle with a poor version. The subcommittee disagreed with his understanding of the function of INDEPENDENT; they saw it as a statement about the behavior of the program, not a construct to force certain behavior. In this vein, they voted against both proposals by counts of 0-4-2 and 1-4-1, respectively. The full committee did not move either proposal for a full vote.

INDEPENDENT comment 3. Extend INDEPENDENT to include reduction operations.

This suggestion had been discussed (albeit with different syntax) at previous HPFF meetings. The problem in the past had been reconciling reduction operations (by their nature an interaction between iterations) with the declared independence of the loop. The subcommittee recommended against this proposal on that basis, and the full committee agreed.

INDEPENDENT comment 4. Clarify what is meant by nesting cases in INDEPENDENT.

Tom Haupt had pointed out some ambiguities in the syntax rules, particularly in deciding how INDEPENDENT assertions could be applied to DO loop nests. His comments boiled down to asking which of the following three cases were legal in HPF:

Case 1
```
!HPF$ INDEPENDENT(I,J)
DO I = 1, NI
   DO J = 1, NJ
      DO K = 1, NK
         ...
```

Case 2
```
!HPF$ INDEPENDENT(I,J)
DO I = 1, NI
   DO K = 1, NK
      DO J = 1, NJ
         ...
```

Case 3

```
!HPF$ INDEPENDENT(I,J)
DO K = 1, NK
   DO I = 1, NI
      DO J = 1, NJ
         ...
```

The committee had decided that case 1 was definitely OK, case 2 was definitely not OK, and case 3 was not intended, but (debatably) allowed by the current syntax. To solve this problem, they recommended restricting the syntax further so that only case 1 was allowed. The exact proposal was that the INDEPENDENT directive (if present) must immediately precede the statement to which it applied and that no index-list would be allowed for INDEPENDENT applied to DO loops. Under questioning, "immediately preceding" was defined as meaning that the next non-comment line must be the DO or FORALL. An index-list was still allowed when the INDEPENDENT applied to a FORALL. Ken Kennedy noted that eliminating the index-list in this way did not preclude adding it back in a later version of HPF. Andy Meltzer moved an amendment to the proposal to eliminate the index-list from all INDEPENDENT directives. After a short discussion, the amendment passed by a vote of 20 in favor, 1 against, and 4 abstaining. The overall motion, that INDEPENDENT must immediately precede the statement it applied to and that there be no index-list, passed by a vote of 20 in favor, 0 opposed, and 5 abstaining.

Line Numbering

Before breaking for lunch, the committee considered one last motion. Matt Snyder from Lahey Computer moved that line numbers be added to the draft. The argument was that implementors needed a finer granularity of reference than just page numbering. Guy Steele replied that due to the structure of TeX, such numbering was not possible without several weeks of work; he suggested, however, that adding a static ruler along the side of each page would be relatively easy. The ruler would be essentially exact line numbering for pages consisting only of ordinary text, but would fall out of synch with the actual lines when the page contained other material (such as figures, code fragments, and section headings). A straw poll was taken on whether to do something about this issue, finding 7 favoring change, 6 wanting to leave the document formatting alone, and most of the room with no opinion. A second straw poll found that, if changes were to be made, 18 favored Guy's ruler and none favored more work than that. Guy promised to write the appropriate macros quickly.

Data Distribution Comments

Guy Steele was the first presenter after lunch, going through the comments on DISTRIBUTE and ALIGN. He noted that there were also many typos that would be corrected silently. All recommendations came from the distribution subgroup, which met the afternoon and evening before. Due to the large number of proposals, noncontroversial changes were accepted without formal votes.

1. Delete all references to "Cheshire cat" and all words beginning with "MELL."

Guy noted that Andy Meltzer had objected to an earlier form of the proposal, which deleted all words beginning with "MEL." The proposal was accepted by acclamation.

2. Get rid of the “index space” terminology and revert to “TEMPLATE.”

The revised terminology was an attempt to keep the concept without using the words that had caused so much controversy. According to public comment, it failed. The proposal was accepted without a formal vote.

3. Forbid duplicate declarations.

These could occur in the context of combined directives such as

```
            !HPF$ DIMENSION(100), DISTRIBUTE(BLOCK) :: a
```

The solution was to delete DIMENSION in the syntax for combined-directive for arrays. The proposal passed without a vote.

4. Provide a way to describe the mappings of components of derived types

The subcommittee recommended waiting for the next round of HPFF.

5. Change PROCESSORS to some other name such as NODES.

The subcommittee reported a unanimous NO vote, taken after much discussion. Several committee members expressed surprise that any subcommittee recommendation could be unanimous.

6. Delete the VIEW directive; use ALIGN instead.

It was clear that the two constructs were closely related anyway. However, the subcommittee voted a unanimous NO (again after much discussion).

7. Impose restrictions on the VIEW directive.

The proposal was based on the observation that “sufficiently devious” chains of VIEW directives can rearrange prime factorizations (of array sizes) arbitrarily, thus making efficient implementation difficult at best. A long subcommittee discussion had produced a concrete proposal:

1. In “VIEW OF p :: q”, q must be one-dimensional.
2. No chaining of VIEW directives is allowed.

The subcommittee recommended this proposal by a vote of 9-0-6. Joel Williamson quickly noted that under these rules a program can’t relate 2-by-4 and 4-by-2 processor arrays, making this a strong restriction. Guy agreed that the proposal was generally very restrictive. A brief discussion suggested restricting VIEW so that the “viewer” must be lower-dimension than the “viewee.” (For clarity, in

```
            !HPF$ VIEW OF p :: q
```

the processor arrangement q is the viewee, and p is the viewer.) In answer to Joel Williamson’s question, “Does this take the heart out of your idea?” Piyush Mehrotra said that the current proposal can’t do his original examples. Alok Choudhary moved to amend the proposal to restrict the viewer only to be lower- or equal-dimension than the viewee. Rob Schreiber further amended the proposal to say that each dimension of “q” be product of several contiguous dimensions of “p.” Before the amendments could be considered, Guy urgently requested tabling the discussion to consider the next proposal.

8. Eliminate VIEW

The subcommittee recommended this option by a vote of 6-5-4. The full committee vote was quickly called, receiving 14 yes, 8 no, and 4 abstaining. The VIEW directive was therefore sent to the Journal of Development. Ed Krall asked if this meant that the

PROCESSORS directive must be modified. The answer was no; the effect was just that there is no defined (HPF level) relation between different-shaped processor arrangements.

9. We *must* fix the significant blanks problem.

This was a unanimous subcommittee recommendation. It did, however, bring up the more controversial question of how to say "ALIGN A WITH B" simply. Several options were presented

```
!HPF$ ALIGN A(:,:) WITH B(:,:)
!HPF$ ALIGN WITH B :: A
```
This received 9 votes in subcommittee.

```
!HPF$ ALIGN A() WITH B()
!HPF$ ALIGN WITH B :: A
```
This received 0 votes in subcommittee.

```
!HPF$ ALIGN A -> B
!HPF$ ALIGN -> B :: A
```
This received 2 votes in subcommittee.

So the subcommittee recommended that the "parenthesis-less" form be retained only in the "::" syntax. Rob Schreiber wanted to amend the proposal so that nonconforming ALIGN directives were disallowed; Guy Steele pointed out this was not legal now. There was no other objection to the proposed solution. Mary Zosel wanted to further restrict the syntax so that blanks were significant even in fixed form directives, thus allowing "dumb parsers" to be built from LEX and YACC. She strongly agreed, however, that syntax should be unambiguous even in the presence of insignificant blanks, and full statements in fixed source form still must obey the usual source rules. To illustrate, she gave two lines of HPF.

```
!HPF$ ALIGNA(:)WITHB(:)

!HPF$ ALIGN A(:) WITH B(:)
```

(Note the blanks in each line.) Both lines would be allowed by the subcommittee recommendation, but only second by Mary's proposal. A formal vote to require significant blanks in directives only netted 9 yes, 4 no, and 12 abstain.

10. Are blanks between keywords mandatory in free source form (i.e. ALIGN WITH, etc.)?

The subcommittee responded "yes" by a unanimous vote. The full committee agreed.

11. Change full HPF rules for forms of align-subscript to be same as the rules for Subset HPF.

12 Allow at most one occurrence of an align-dummy in an align-subscript.

13. Replace Subset HPF rules for forms of align-subscript to be same as rules for full HPF, as modified above.

The subcommittee recommended against proposal 11 (no vote given), and in favor of proposals 12 (vote 11-0-4) and 13 (13-1-1). The net effect of these recommendations was to make full HPF more restrictive (eliminating complex expressions like 5*(j+1)-2*(j+1)-1) and Subset HPF more liberal (allowing expressions like 3*(j+1)-1). The issue was presented as one of allowing simple preprocessors and program generators to produce HPF. The vote to restrict the full language was 11 yes, 4 no, and 10 abstain. The vote to make Subset HPF equivalent (for this issue) to full HPF was 16 yes, 4 no, 5 abstain.

14: Delete the colon notation from ALIGN.

In this case, “the colon notation” referred to forms like

```
!HPF$ ALIGN a(:) WITH b(:)
```

rather than the double-colon Fortran 90 notation for declarations. The reasoning behind the proposal was that this notation duplicates the usual align-dummy functionality; in part the original comment had read “Simplify, simplify, ...” The subcommittee recommended against this proposal by a vote of 2-7-5. The proposal was moved but not seconded in full committee.

15. Forbid mixing of “:” and named align-sources in a single align-source-list.

Again, the reasoning was to simplify the expressions. The subcommittee recommended against by a vote of 3-7-5. The arguments against the proposal were by analogy with array expressions and FORALL, which both allowed mixing. A full vote was taken, producing 1 yes, 18 no, and 3 abstaining.

16. Correct BNF and constraints: each align-dummy should appear in at most one align-subscript.

The reviewer had observed that , the BNF allowed J*J as a linear function of J. The proposal was accepted without debate.

17. In Subset HPF, omit the INHERIT attribute and directive and don’t allow a dist-format-clause or dist-target to be transcriptive (i.e. a “lone star”).

The subcommittee recommended in favor, voting 8-2-2. Their reasoning was that this requires as much work as DYNAMIC distribution features, which are not in Subset HPF. Clemens Thole wanted to amend the proposal to split it into two votes: omit INHERIT as one vote, and no transcriptive features as another. After some discussion, the proposal was amended so that in Subset HPF, INHERIT requires an explicit DISTRIBUTE. (This avoided the difficult implementation cases.) The vote was 9 yes, 2 no, 11 abstain. Someone commented that “This vote justifies the t-shirt” Rewording the vote as “In Subset HPF, don’t allow a dist-format-clause or dist-target to be transcriptive, either explicitly or implicitly” produced a more definite result of 15 yes, 2 no, 6 abstain.

18. Change the meaning of BLOCK not to be origin-anchored.

The proposal needed an illustrative example:

```
REAL a(100), b(100)
!HPF$ DISTRIBUTE (BLOCK(10)) [ONTO p ] :: a, b
a(1:90) = b(11:100)
```

The question was, “Is the compiler free to pick the origins of processor arrangements?” In the example, it would clearly be advantageous to choose different offsets for arrays a and b to line the blocks up on the same processors. The subcommittee decided that, with the ONTO clause present, the answer was definitely “no.” Without ONTO, the subcommittee thought the compiler should be allowed to pick offsets. Several people, including Ken Kennedy, observed that since the directives were only advisory anyway, the compiler could ignore them. Rob Schreiber thought this was a terrible idea, since the user can express exactly the behavior that was wanted by alignment to a TEMPLATE. Richard Swift proposed that, absent an explicit ALIGN, the programmer would need to assign processor offsets. However, with the demise of VIEW there was no way to do this. This led to a long discussion about "How do we know that processor numbers relate to anything?" It was finally decided that HPFF needed to make the document clear: Can processor arrangements be offset from other? This was not allowed at the programmer level, but according to the proposal the compiler should be able to do so. David Presberg generalized this to "Should the compiler have leeway to perform optimizations that users cannot?" Much more discussion followed. Andy Meltzer summed up his final position on the issue as "I was wrong, but that's because I was right yesterday - I turned out to be confused anyway." Finally, there was a call for votes: Should HPF add this difference in behavior to the document (thus giving the compiler additional flexibility, "if you believe it does not have it already") The results were 6 yes, 10 no, 7 abstain, and the proposal failed.

19. On page 25, last constraint: Leave the default distribution to the compiler.

The document assigned BLOCK as the default; the subcommittee recommended removing this constraint by a vote of 8-0-3. The proposal passed without vote.

20. Provide a user-specified default distribution (actually 7 distributions, one for each possible rank).

The proposal even gave an example syntax:

```
!HPF$ DISTRIBUTE (CYCLIC) ONTO allprocs
!HPF$ DISTRIBUTE (CYCLIC,CYCLIC) ONTO xy_procs
!HPF$ DISTRIBUTE (CYCLIC,CYCLIC,*) ONTO xy_procs
```

A comment from the floor said that MasPar found its users want this feature. Rob Schreiber described it as the moral equivalent of IMPLICIT. The proposal was moved and seconded. Clemens Thole asked how this related to COMMON blocks, and was told that COMMONs must already be declared consistently. A similar question relating this to the to SEQUENTIAL directive put the default at the same level, basically providing an override for the rank of arrays given. A straw poll asked if the subcommittee should draft a tighter version of proposal; 9 voted yes, 13 voted no, and 6 abstained.

21. Allow empty dist-format meaning "compiler chooses."

For example, the following would be legal:

```
!HPF$ DISTRIBUTE a(BLOCK,,*,)
```

Don Heller said he didn't like blanks used like this. Alok Choudhary suggested using "?" instead of no option. The fundamental question, however, was "Do we want this capability?" A straw poll to consider this proposal further showed 5 in favor, 14 opposed, and 8 abstaining, so the proposal died.

22. Delete DYNAMIC, REALIGN, and REDISTRIBUTE from HPF.

This elicited remarks including

"Oh, no, another MasPar proposal!"
"If you guys will just go to MIMD machines..."
"If you had a message-passing machine, boy would we have a committee for you." (referring to the MPI forum)

Guy Steele ended the discussion with “Enough ad hominem and ad corporatem attacks” The vote to remove the features failed 5-18-0.

The group then took a well-deserved break.

Intrinsics and HPF Library Comments

Rob Schreiber presented the public comments on intrinsic functions and the HPF library. Like the other groups, he started his presentation with disclaimers about typographical errors and formatting concerns. All recommendations came from the intrinsics subcommittee, who had met the previous evening.

1. Remove the EXCLUSIVE option from COPY_XXXFIX.

The affected routines were

```
copy_prefix( (/ 1,2,3,4 /), exclusive = .TRUE. )
copy_suffix( (/ 1,2,3,4 /), exclusive = .FALSE. )
```

Public comment had noted that there was no semantic difference between EXCLUSIVE=.TRUE. and EXCLUSIVE=.FALSE. for these routines, and recommended dropping the option. It had apparently been included only for compatibility with other prefix and suffix routines. The proposal was accepted without dissent.

2. Change the library module name from HPF_LIB to HPF_LIBRARY.

The argument was that this was a clearer name. The subcommittee recommended in favor of the proposal, and it was accepted without dissent.

3. Reorder the sections in this chapter.

The proposed order was

1. System inquiry functions.
2. Mapping inquiry (HPF subroutines).
3. Distribution inquiry (local routines).
4. Computational intrinsics.
5. Library procedures.

The subcommittee agreed this was a better ordering, since it grouped inquiry routines and intrinsics in intuitive ways. It was accepted without dissent.

4. Change the CHARACTER output arguments of inquiry functions to have unspecified lengths.

These arguments were limited to 10 characters, which the reviewer claimed was limiting to future extensions. The proposal would change this to truncate returned arguments if storage was not available, or pad the result with blanks if it were assigned to a longer string. This was in conformance with standard Fortran 90 practice. The change was recommended by the subcommittee and accepted without dissent.

5. Make the new intrinsics available through a module as USE HPF_INTRINSICS.

A more detailed wording of this proposal was

```
If X3J3 allows additional intrinsics in MODULES;
    When MODULES are implemented in HPF, allow “USE
        HPF_INTRINSICS”
    Now allow “!HPF$ USE HPF_INTRINSICS”
```

The wording was complicated by the fact that X3J3 was in the process of interpreting a query to allow such modules; thus, the proposal made sense if X3J3's response was "Yes" but confusing at the moment. The reason given for this proposal was to remove ILEN and the other new intrinsics from the default namespace; if intrinsic modules were allowed, it was assumed this would become standard practice. This sort of MODULE / name stuff was coming in because there were lots of proposals for language bindings, and namespace pollution was becoming a problem. The Fortran 90 standard does not obviously preclude this treatment, and the interpretation got support at the last X3J3 meeting. It was suggested that HPF should make the USE statement required, pending the X3J3 decision. Richard Swift moved that the proposal be put in the Journal of Development, since the X3J3 decision wouldn't come before HPF 2 anyway. He did favor incorporating it then. A vote was taken, reporting 18 yes, 0 no, and 6 abstaining, so the proposal was tabled until the 1994 round of HPFF.

Non-technical Matters

A short discussion generated a list of topics for discussion the next day:

- Extrinsics
- Data parallel leftovers
- Data distribution leftovers
- Journal of Development handling
- Final draft / editorial work schedule.
- Support tools discussion
- Supercomputing '93 tutorial & workshop planning
- Vendor support / announcements (also known as the "vendor disclaimer session")
- HPF follow-on planning
- Technical meeting planning (in summer 1993)
- Low Performance Fortran

The group agreed to tackle the little stuff now, if possible. For the most part, this translated into the non-technical planning items.

Robbie Babb agreed to serve as the Journal of Development editor, solving the problem of who to send those comments to.

There had been several requests for a vendor poll to determine who was implementing HPF. After a long discussion of what information the group could gather, two lists were developed.

Names of Publicly Announced Efforts

- ACE
- APR
- DEC
- Intel
- Kuck and Associates
- Lahey
- MasPar
- Meiko
- NA Software
- Portland Group
- Pacific Sierra Research
- Thinking Machines

Other Active Participants in HPFF

- Convex
- Cray Research
- Fujitsu
- HP
- IBM
- Meiko
- nCUBE
- Sun

An attempted statistical summary of vendors who had not made formal announcements found 0 saying they definitely will implement HPF, 1 “maybe,” and 4 “no comment.”

Alok Choudhary had previously volunteered to organize an HPF workshop at Supercomputing ’93 (in Portland). The purpose would be primarily to get user feedback on HPF, based on the expected products appearing late in the year. The workshop would not be a tutorial on HPF, but more of a panel discussion. He circulated a proposal based on David Loveman’s from SC ’92, which the committee thought was generally good. Volunteers were needed to organize the meeting, and the issue of whose name should appear on the proposal came up. The group agreed to have the workshop officially organized by “HPFF committee, Ken Kennedy chair,” and Chuck Koelbel and Mary Zosel volunteered to contribute to the planning. On the question of “Who will participate?” John Levesque and Thinking Machines volunteered. Vince Schuster of the Portland Group, Incorporated noted that they would also be “in the area” during the conference. The organizers would have to search for HPF users to form the remainder of the panel, but their names were not needed on the preliminary proposal.

The HPF tutorial had previously been suggested but not planned. Some on the committee suggested a combined Fortran 90 / HPF tutorial; others expressed doubts about the time needed for both tutorials. Rob Schreiber had the last word, saying the HPF tutorial should run stand-alone. Ken Kennedy suggested scheduling the HPF and (expected) Fortran 90 tutorials to avoid conflict, and this was added to the list to be included in the proposal. Rob Schreiber volunteered to put the HPF tutorial together.

The summer meeting had been envisioned as an implementer’s meeting to exchange ideas and information. Another purpose would be to handle interpretations of the HPF document. Planning of this was delayed until a firmer idea of the amount of work could be set. However, the mention of interpretations led to a discussion of new mechanisms for feedback.

The group quickly decided that a new hpff-interpret mail alias would be a good idea, but momentarily floundered on the question of “Who’s in charge?” Andy Meltzer noted that determining which chapter was involved in a question is easy, and suggested forwarding the questions to the chapter editor for answering. Jerry Wagener described the X3J3 f90-interp reflector, which was an initial gathering point for all questions. Usually a question would draw several comments, and if anyone thought the issue was important then it would be sent in as an official request for interpretation. Many questions, however, were quickly answered by pointing the questioner to the right section of the standard. Mary Zosel volunteered to keep a list of outstanding issues, and Chuck Koelbel promised to create the hpff-interpret mail group (at cs.rice.edu, like the other lists) and make it publicly open.

Finally, the group considered the status of the test suite. The Syracuse Fortran 90D test suite was already available, but HPF versions of the programs had not been developed for lack of a compiler. The committee immediately asked if the HPF test suite could be created now, receiving a “yes” answer. Tom Haupt would ask for new tests in an announcement letter, and make the first set of HPF benchmarks available by May. These programs would use the FORALL construct, basic distribution mechanisms, and stay within Subset HPF. The Syracusians politely declined to guarantee that they would eventually test all Fortran 90 features in Subset HPF, but did believe it would exercise a substantial portion of HPF. It would certainly be enough to test initial versions of compilers. After a short discussion of the role of the test suite (and the constraints on the Syracuse group), the group accepted this and broke for dinner.

Low Performance Fortran

Friday started with Andy Meltzer's report on Low Performance Fortran.

Next LPF Meeting

- Hawaii
- To attend:
    1. Send $1000 to Andrew Meltzer, 795 Holly Ave. #1, St. Paul, MN 85104
    2. Buy yourself an airline ticket
    3. Make hotel reservations

(Asked "Which island?" Andy assured the group that any island would be fine. Also, to keep overhead low, LPFF would not send any confirmation notices.)

LPF Feedback

- As you recall from a decision in an earlier meeting, all LPF users are to be shot.
- This has been extended to reviewers.
- Got 12 responses.
  11 people have been shot.
- One review was anonymous:
  Suggestion: Forget the whole thing.
  Proposal accepted.

LPF Funding Status

- Income:
  Found between sofa cushions: $0.18
  Petty theft: $3.22
  Blackmail: $12.00
  NSF grant: $45,000.00
  Total: $45,015.40
  (Blackmail - threatening to reveal that a person was involved in LPF - is expected to be a future growth area.)
- Expenditures:
  Bribes to NSF: $5,000.00
  Entertainment of funding agencies: $40,000.00
  Clothing: $15.00
  Total: $45,015.00
- Balance: $0.40

LPF Tutorials

- 2 planned
    1. Kjokkeran, Finland in late December
    2. Oagadougou, Burkina Faso in June

LPF Announced Vendors

- RCA
- SSI
- FPS
- Raytheon
- Bendix
- PRIME
- Honeywell
- Osborne
- SAXPY
- BBN

LPF Maybe Vendors

- Cray Research
- IBM
- DEC
- Thinking Machines
- Alliant
- Archipel
- Convex
- Fujitsu
- HP
- Intel
- Lahey
- MasPar
- Portland Group

Finally, Andy announced the results of the LPF t-shirt competition. The first two t-shirts were the most in the spirit of LPF, so they were rejected (recall that any good suggestion to LPF is automatically rejected). Andy showed pictures of a pair of boxer shorts and a tennis shoe, both with “LPF” in large letters. He then threw off his flannel shirt to show the real LPF shirt - a plain white t-shirt. “No, just kidding.” He removed that shirt to reveal - the HPF t-shirt. After a round of boos, he stripped that shirt off to reveal - the Cray Research compiler group t-shirt. But that would get him in too much trouble, so he removed that shirt to reveal the final, official LPF t-shirt. On the front was a black dot with an arrow looping from the bottom of the dot to the top; on the back was the slogan “Don’t blame us - We don’t have that feature!” Andy accepted the applause, deflected the t-shirt orders, and took his seat.

After Andy’s presentation, Rob Schreiber, Mary Zosel, Guy Steele and David Presberg “suggested some sites for the next LPF meeting” by performing the Geographical Fugue. This piece of music is performed a cappella consists entirely of the names of cities and other locations around the world.

More FORALL Comments

Chuck Koelbel reported on the FORALL subgroup’s discussion of “the elemental question.” The discussion had gone on late into the night. A final vote on whether to change the current draft had favored leaving things as they were, by a vote of 4-5-0. Given the closeness of that vote, two more polls were taken in subcommittee: “If we change anyway, should we add user-defined elemental functions?” was accepted by a vote of 5-0-4. It was noted that a proposal to this effect was already in the Journal of Development. The subcommittee vote on “If we change, should we replace pure by elemental?” was against changing (3-6-1). The subcommittee therefore recommended not adding elemental functions to HPF. After a short discussion, it was clear that nobody was going to change their minds on the issue, although many were willing to present arguments on both sides. Due to a gentlemen’s agreement the night before, the subcommittee members were reluctant to move the issue for a vote in full committee; however, a motion was eventually made to add elemental to HPF, with the proviso that the vote be taken without discussion. The vote was 8 in favor, 8 opposed, and 6 abstaining. The motion therefore failed. In light of the importance of this issue, Chuck Koelbel and David Loveman moved that HPFF send a letter to the ANSI X3J3 committee strongly recommending that they consider elemental functions for the next Fortran standard. This motion passed with 22 in favor, and none opposed or abstaining.

Chuck then noted a few other proposals from the subcommittee meeting. First was a “last impassioned plea for nested WHERE constructs.” Guy Steele had sent this as a note to the FORALL committee while he was chairing the distribution group. The FORALL subcommittee recommended against this proposal by a vote of 1 in favor (cast in absentia), 4.5 opposed, and 1.5 abstaining.

The last slide was a warning from the subcommittee: “We didn't understand the relationship between POINTER and DISTRIBUTE.” Two examples were given: a POINTER must be resolvable on all processors (some subcommittee members assumes this was obvious, some were taken aback), and a troublesome case of alignment:

```
REAL, POINTER, DIMENSION(:) :: p
REAL, TARGET, DIMENSION(:) :: x
!HPF$ DISTRIBUTE x(BLOCK)
!HPF$ DYNAMIC p
!HPF$ DISTRIBUTE p(CYCLIC)
...
ALLOCATE(p(100)) ! p is distributed cyclic here
...
p => x ! Is this legal?
```

The fundamental question was whether a pointer must be aligned with its target in pointer assignment. Guy Steele believed that it was consistent to allow DYNAMIC pointers to inherit the alignment and distribution of their targets, but wanted to study the issue before making a firm commitment.

EXTRINSIC Comments

Guy Steele presented the public comments on EXTRINSIC routines. All recommendations were the from the EXTRINSIC subcommittee, which had met the previous afternoon.

1. “Demote” local routines written in HPF, and the inquiry intrinsics used in that mode, to optional status in full HPF.

The subcommittee recommended this proposal by a vote of 7-0-2. Ken Kennedy noted disturbing policy implications in the proposal, which led to a discussion of similar precedents in other standards. Alok Choudhary asked about the portability implications of these features, to which Rob Schreiber responded that HPF can’t implement MIMD algorithms on SIMD machines. Chuck Koelbel asked why the local paradigm should not be in Subset HPF. Guy Steele explained that while many vendors need or want such things, some can’t do it, so removing the feature from the language was a compromise to protect an important minority. Richard Swift claimed it was right that machine/class-dependent stuff shouldn’t be in a portable language, but also right that it’s needed. He therefore suggested moving it to an appendix saying “If you do it, spell it this way.” The discussion quickly grew confusing, and Guy Steele suggested tabling it pending the next few points of discussion.

2. What of COMMON or global variables is seen by a local routine?

The subcommittee answered, “There are none.” Such variables belong to the (global) HPF world; local routines belong to another programming model. Vendors are, of course, free to define a compiler-dependent interface for converting such variables. The full committee assented to this interpretation.

3. Why have both EXTRINSIC and LOCAL? Why have LOCAL at all?

The subcommittee answered that an interface to other programming models was needed. The rejoinder was “But aren’t there more than two models?” to which the committee responded “Right.” They therefore proposed to replace “EXTRINSIC” by “EXTRINSIC(model_name)”. “LOCAL” would then become a model name (the only standard one). Moreover, EXTRINSIC is syntactically like PURE (or RECURSIVE), a semantic attribute rather than a directive, and should be made first-class syntax. For example,

```
interface
   extrinsic(local) function bagel(x)
   real x(:)
   end function
end interface

extrinsic(local) function bagel(x)
   real x(:)
....
   return
end function

interface operator(.grotesque.)
   extrinsic(local) function foo(a,b)
   ...
   end function
   extrinsic(spmd_forth) function foo(a,b)
   ...
   end function
end interface
```

This shows the interface declaration and procedure definition of an EXTRINSIC function bagel, and two implementations (in different programming models) of an operator named grotesque (some declarations of a and b would be needed to resolve the overloading. The subcommittee had also considered naming the model; a straw poll had favored SPMD (5 subcommittee votes), over LOCAL (2), and none of the above (3).

Andy Meltzer argued that these features would be non-portable, since LOCAL code is inherently processor-specific. This led to a discussion of what should be part of HPF and what should not. Rob Schreiber and David Loveman argued that nonportable features could be included by analogy to KIND type parameters - you can ask for a KIND with 18-digit precision, but the machine may not support you. This led to more support for moving LOCAL into an annex, as proposed earlier. A brief discussion of how specific the synchronization requirements were to the LOCAL model produced the answer “sort of” - the wording that an EXTRINSIC “must behave as if the subroutine completes before return" may need some rewording. The technicalities of this wording were relegated to the author and subcommittee.

Finally, a vote was called on the whole package: move the LOCAL model to an annex, change the spelling to EXTRINSIC(model_name), and make the EXTRINSIC chapter contain the interface definition. The vote found 18 in favor, 1 opposed, and 2 abstaining.

Next there was a discussion of whether EXTRINSIC should be a directive or real syntax. Richard Swift claimed that directives worked fine, but Rob Schreiber argued that it changes the semantics of a program since normal calling sequences won't work. The vote to make EXTRINSIC into syntax passed with 7 yes votes, 4 no votes, and 9 abstentions.

Finally, the committee considered what the standard model name should be. Initial nominees included LOCAL, SPMD, HPF_LOCAL, NODE_HPF, and no name (using the standard model as a default). After a spelling discussion, HPF_LOCAL was moved as the name. The vote passed 18-0-4. A further motion to make the model_name optional (with HPF_LOCAL as the default) started a short debate on the wisdom of defaults in general. It ended with a vote of 3 in favor of this default, 13 opposed, and 6 abstaining.

Now that the entire syntax had been defined (technically, as a series of amendments), a final vote to accept the full proposal passed by a count of 16 yes, 0 no, and 6 abstain. A final motion to reserve "HPF" as the model name for our model (i.e. for use when calling HPF modules from other paradigms) passed 15-1-6.

Editorial Discussion

David Loveman led a short discussion of editorial matters for the final draft.

1. New macros.

Guy Steele promised new macros to automatically number syntax rules and generate the text ruler automatically. He warned that these might change other formatting slightly, for example by squeezing space available for syntax rules. Also, "Chapter" must be changed to "Section" in the text as well as in the macros.

2. Text correcting

David implored the chapter authors to base their corrections on the DRAFT text, not the original text (which may be different, due to formatting changes and spelling corrections). Two of the larger text changes involved the Journal of Development. First, it became a new document, requiring a reference in Section 1 and the bibliography. Then the material in Section 7 had to be reworked for inclusion in that document, as well as a shorter version inserted in Section 1. Chuck Koelbel promised to do both changes. David also reminded the group that the sooner corrections are done, the sooner version 1.0 is done.

This led to a discussion of how to keep the public informed of the work on the new draft. Chuck Koelbel agreed to send a summary of changes from this meeting to the hpff mailing list. Next was the question of what to call this version. The current version was labeled as "1.0 draft," so the group felt that just "version 1.0" was not sufficiently different given the changes made. Guy Steele pointed out that "There is precedent for the first widely available version of a language to be numbered 1.5." However, the group did not feel this was a good plan either. They finally agreed to call the new specification "version 1.0 final" and change the file name to "hpf-v10-final.*". Deleting all previous versions of the draft from the electronic repositories should eliminate confusion over which was the correct draft.

Timetables

Next came a discussion of the time frame for producing the final draft. Guy Steele volunteered to have the new macros ready by the next Wednesday. Based on the time commitments of the various authors, two weeks seemed reasonable for producing the required rewrites. Working from those dates, the schedule that was finally agreed to was

- March 17: New macros available.
- March 26: New sections from all authors sent to David Loveman.
- April 2: Review copies of whole specification sent to committee.
- April 16: Reviews of new specification send back to David Loveman.
- April 23: Corrections based on reviews integrated by Loveman, sent to chapter editors for final checking.
- April 28: Chapters returned to Loveman.
- May 3: Final version of specification available on-line.

David Loveman sent a sign-up list for reviewer volunteers around the table, and promised to add Michael Metcalf (who had provided voluminous comments previously) as well. All edited chapters would be sent back to the chapter editors as quality control, to avoid over-editing and other sins.

4. No definition of HPF-conforming program - i.e. what is defined?

The committee still had not agreed whether ignoring a directive meant “don’t even check the syntax of directives” or “check syntax, but don’t implement it as defined.” The consensus was to let the compiler market decide what a “real HPF compiler” means. Furthermore, the specification should use “HPF-conforming” and “Subset-conforming” to describe programs as in the Fortran 90 standard. Chuck Koelbel promised to adapt the definitions found there for use in the HPF document.

Odds and Ends

The meeting ended with a few loose ends from the previous day. First, Chuck Koelbel promised to send out notes describing the changes to the draft and the schedule for producing the final version. He also promised to make a best effort on getting the December and March notes in distributable form by the time the draft appeared.

Having completed the work at this meeting, the group decided that no April meeting was needed. (One had been tentatively scheduled in case of more comments than could be handled.)

A few ideas were kicked around for the HPFF 2 inaugural meeting next January. Some questions were raised about what should be considered in HPFF 2, with the general answer that no a priori limitations should be set. As a format for the meeting, inviting HPF users for experience talks got general support. It was expected that vendors and HPF non-users would also have some opinions on what was missing from the first version of HPF. Ken Kennedy, Chuck Koelbel, and others in the group proposed to do more detailed logistical planning in time to announce plans at Supercomputing ‘93 (or before). A motion was made to change the meeting place for both the kickoff meeting and (maybe especially) the technical meetings, but no hard decisions were made regarding where.

Two final technical questions were discussed. First, the question of applying distribution intrinsics to sequential arrays was raised. In particular, the question was what such functions should return. Rob Schreiber suggested that HPF should forbid that case, and Guy Steele quickly agreed. The rest of the committee followed suit.

The last question was the treatment of ALIGN applied to a POINTER used in pointer assignment, as the FORALL group had mentioned. Guy Steele proposed that if a POINTER variable appears in any pointer assignment left-hand side, then that variable may not be an alignee in ALIGN or distributee in DISTRIBUTE. He further pointed out that REALIGN and REDISTRIBUTE do not cause trouble in this regard, since HPF already has restrictions on memory pointed to by two paths. Piyush Mehrotra proposed a more liberal solution in which ALIGN or DISTRIBUTE of a POINTER only describes the distribution of the array when it is ALLOCATEd. The declarations do not apply when the pointer is reset by pointer assignment. After a few questions about the consistency of this, the committee was convinced that Piyush's solution was workable. It was therefore accepted, and Guy promised to write it up as part of the final draft of the language specification.