I.P. Sharp Associates

LEVEL III

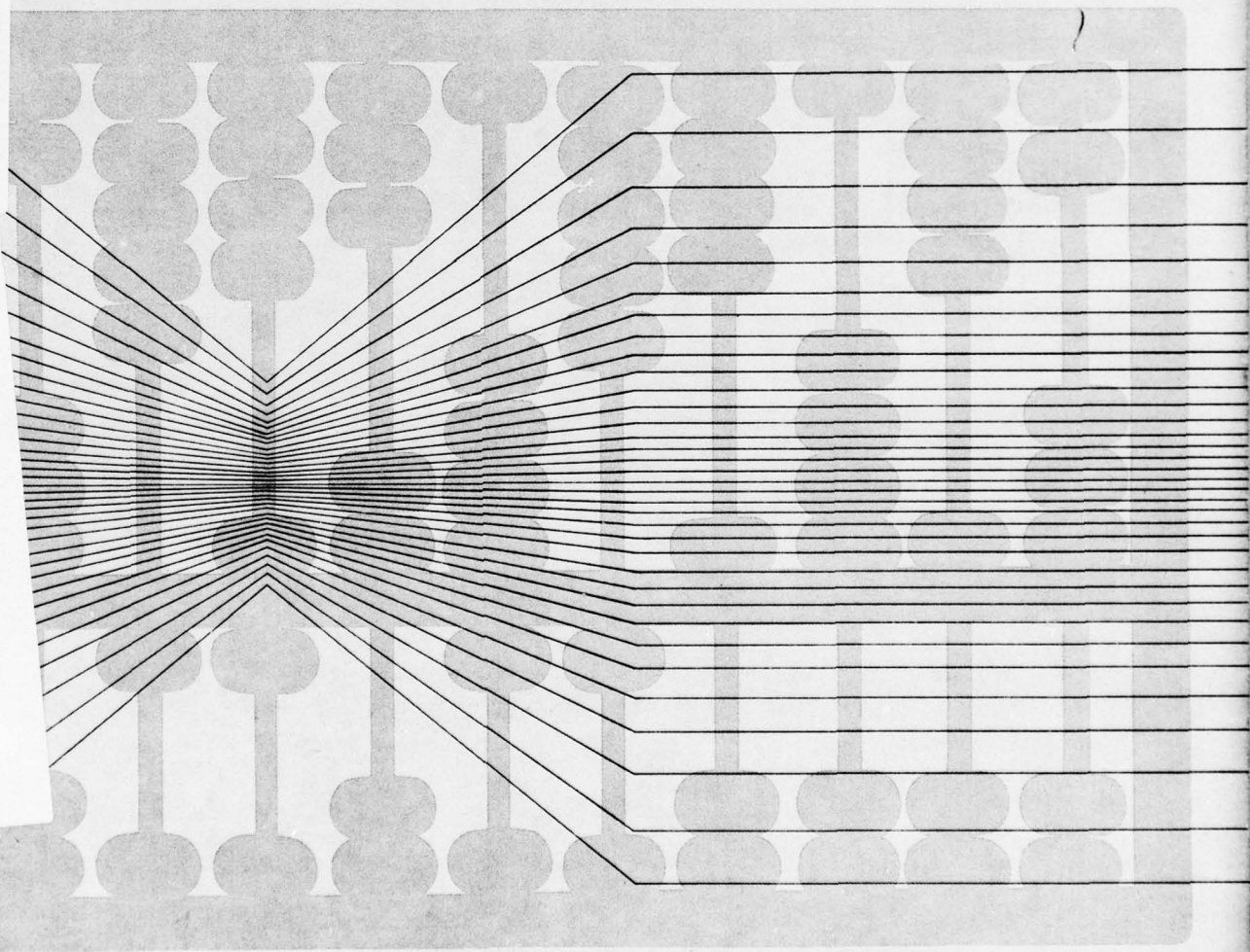
12

AD A061052

AD A061402

DDC
RECEIVED
OCT 21 1978
DISSEMINATED
F

DDC FILE COPY



This document has been approved for public release and sale; its distribution is unlimited.

78 11 21 044

Quarterly Technical Report
for

EUCLID Compiler for PDP-11
Number 1

PERIOD COVERED: 1 October 77 to 31 March 78

This research was sponsored by the
Defense Advanced Research Projects
Agency under ARPA Order No. 3475
Contract No. MDA 903-78-C-0037
Monitored by Steve Walker
Effective Date of Contract 1 Oct 1977
Contract Expiry Date 31 Mar 1979

A portion of this project is being
sponsored by the Canadian Department
of National Defence, Chief of Research
and Development

David Bonyun
I.P. Sharp Associates Limited
Suite 600
265 Carling Avenue
OTTAWA, Canada K1S 2E1

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE *(and Subtitle)*

(16) EUCLID Compiler for PDP-11, Report Number 1

5. TYPE OF REPORT & PERIOD COVERED
Quarterly Technical to March 13, 1978

6. PERFORMING ORG. REPORT NUMBER
(17) IPSA-3819-001

7. AUTHOR(s)
D.A. Bonyun, I.P. Sharp Associates
R.C. Holt, University of Toronto

8. CONTRACT OR GRANT NUMBER(s)
(15) MDA-903-78-C-0037, ARPA Order-3475

9. PERFORMING ORGANIZATION NAME AND ADDRESS
I.P. Sharp Associates Limited
600-265 Carling Avenue
OTTAWA, Ontario, Canada K1S 2E1

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Information Processing Techniques Office
DARPA ARLINGTON, VA 22209

(11) 12. REPORT DATE
21 April 1978

13. NUMBER OF PAGES
29

14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*
(10) David A. Bonyun
R.C. Holt

15. SECURITY CLASS. *(of this report)*

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

(12) 31 p.

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

(9) Quarterly technical rept. no. 1.
1 Oct 77-31 Mar 78,

18. SUPPLEMENTARY NOTES

Part of this project is funded by the Canadian Department of National Defence

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

EUCLID, compiler, computer security

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

The work towards a EUCLID compiler for the PDP-11 is proceeding satisfactorily although it has changed somewhat as the result of effort put into stabilizing the language.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

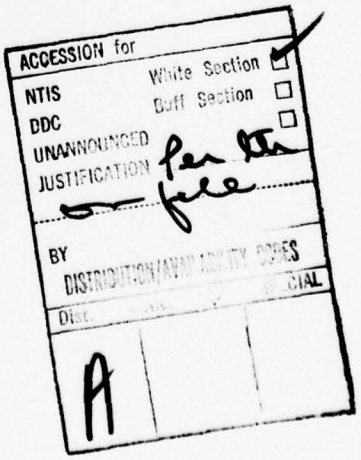
UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

392 956

Quarterly Technical Report #2
EUCLID Compiler Project

Table of Contents



Quarterly Technical Report #1

EUCLID Compiler Project

Report Summary

The purpose of the project is to produce a compiler for the language EUCLID for the PDP-11 computer. During the six months of activity since 1 Oct 1977, the implementation team at the University of Toronto have completed a "transliterator" which takes a well-defined subset of EUCLID to the UNIX language, C. This product was delivered in January to FORD/SRI and to TRW, the two potential users of EUCLID in the development of KSOS, the Kernelized Secure Operating System.

In addition, within the reporting period, much work has been accomplished in stabilizing the language and in preparing the second product, the "translator". This will compile full EUCLID to PDP-11 assembler omitting one or two of EUCLID's particular difficulties especially the handling of legality assertions.

The techniques employed by the team are essentially those pioneered and developed by the University of Toronto's Computer Systems Research Group. The team is composed of two professors (Holt and Wortman), one CSRG staff programmer (Cordy) and two I.P. Sharp people (Crowe and Bonyun). The strategy is to move through three distinct but unequal phases, each with a product (the transliterator, the translator, and the final compiler).

The work is progressing smoothly and satisfactorily except that it has fallen behind schedule by one to three months. The reason is that a great deal of unexpected effort has gone into stabilizing the language which was not as well-defined and bug-free as had been anticipated when the project began. Appendix C is a catalogue of communications with the EUCLID committee on these matters. Appendix D is the printed form of the minutes of a meeting between the implementation team and the EUCLID committee at ZEROX PARC in January 1978.

Several working papers have been produced. These are catalogued in Appendix E.

Quarterly Technical Report #1

EUCLID Compiler Project

I. BACKGROUND

This work was undertaken at the suggestion of Mr. Steve Walker, formerly of the Information Processing Office of ARPA, who had caused the language EUCLID to come into being originally. The language is designed to permit the coding of programs to be verified. This was specifically done to permit the coding of the security kernel for the UNIX operating system.

Because of the desire of the Canadian Department of National Defence (DND) to contribute to and to participate in the work towards a fully secure, kernelized operating system, the initial stage and some of the intermediate stages of this project are being funded by them. The work is being done in Canada by a team composed of people from I.P. Sharp Associates Limited and from the University of Toronto. It is being monitored, in part, by DND.

II. THE APPROACH

The work being undertaken follows exactly the plan laid out in the original proposal. Essentially this involves the delivery of three separate products: a transliterator, a translator and the final compiler.

The transliterator takes a well-defined subset of EUCLID, dubbed "small EUCLID" to the UNIX language, C. The subset was

chosen on two bases: what was required for the compiler (which is to be written in the subset), and what was easily translated to C. This transliterator, which permits bootstrapping the compiler, was completed in 1977 and delivered, at the request of Mr. Walker, to TRW at Los Angeles, and to FORD/SRI at Menlo Park. These two are the competetors for the secure UNIX project. Both agreed that the transliterator delivery was successfully accomplished.

The translator is the full compiler omitting one or two of the more complex attributes of the language. In particular the handling of legality assertions will not be considered until the full compiler state. It translates the full language EUCLID into PDP-11 assembler. Although it was to have been delivered in April 1978, language problems (to be discussed in paragraph III) have delayed it until July 1978.

The compiler will incorporate all those features required by the language but deferred from the translator. It is scheduled for delivery by January 1978, but will likely be 1 to 3 months late.

The technology employed throughout the three major stages identified above is that in constant use by the Computer System Research Group of the University of Toronto. They have been successful in a number of previous compilers using the same basic approach, and their familiarity with EUCLID (through Jim Horning who was at U of T while being part of the

design committee) made them obvious choices to do the work. The participation of I.P. Sharp is to provide a commerical basis for continuity and modifications.

III. LANGUAGE PROBLEMS

As has been mentioned above, there have been a number of difficulties detected within the language as the work proceeded. These "instabilities" have been the cause of a great deal of ARPANET communication between the compiler team and the design committee. The communications culminated in a joint meeting in Palo Alto in January 1978 in which most of the outstanding issues were resolved. Appendix C is an index of the ARPANET mail on this subject; Appendix D provides the minutes of the meeting. It is hoped that a new revised language report will appear soon which will set into context all the changes which have occurred.

As an example of issues which have been raised and, for the most part, satisfactorily resolved, there is the problem of the "unspeakable assertion". Although the translator will not handle assertions, it is clear that an eye must be kept on the whole list of different features at this stage so that their subsequent addition will be neither unduly difficult nor impossible.

An "unspeakable assertion" is an assertion occurring in a module and requiring for its statement the use of object names which have not been explicitly imported to that module. The

solution to the problem involves a closure to the imports list.

The language instabilities have caused the team a great deal of unanticipated work. For this reason the translator and the compiler are both likely to be delayed somewhat: the translator is about 3 months late; the compiler's delay is not yet determinable, but will likely not be more than 3 months.

IV. WORKING PAPERS

Throughout the project a number of working papers have appeared and continue to be written. Appendix E gives an index of those appearing within the reporting period. If anyone requires the full paper, requests ought to be directed to David Bonyun, I.P. Sharp Associates Limited, Suite 600, 265 Carling Avenue, Ottawa K1S 2E1, Canada.

V. CONCLUSIONS

The work continues to go satisfactorily. Although the latter two products will be a little late, the prime users of these products do not seem to find a great deal of hardship in the delay. Summaries of work accomplished so far, as seen by the University of Toronto subcontractors, occur as Appendices A and B.

APPENDIX A

Progress Report No 1
(1 Oct. - 31 Dec., 1978)

This period has been occupied in preparing for the construction of a Translator for the full EUCLID language. The central aspect of this work has been the production of a Small EUCLID Transliterator, which will be one of the primary tools for building the Translator. The following has been accomplished during this period.

1. Design and documentation of the Small EUCLID language, which is a subset of full EUCLID. The full EUCLID Translator will be written in Small EUCLID.

2. Maintenance of the Scanner, which makes up part of the Transliterator and later part of the Translator.

3. Design and implementation of the Parser skeleton, which makes up part of the Transliterator and later part of the Translator.

4. Design, implementation, bootstrapping and distribution of the Transliterator, which maps Small EUCLID programs into C programs, so they can be run under the UNIX operating system.

5. Design and implementation of a preliminary I/O support system under UNIX.

6. Extensive interaction with the EUCLID Language Design committee via the ARPANET and at a meeting in California. These interactions have been necessary because various aspects of the language have continued to evolve.

During this period, many of the basic design decisions for the Translator have been made, although detailed design remains to be done. The implementation team has organized itself into an efficiently functioning unit. The EUCLID language has shown

itself to be somewhat more complex and less stable than originally hoped for.

Not as much was accomplished during this period as was hoped for. While this may cause slight deferment of the completion of the project, there seems to be no fundamental difficulty in carrying out the project in approximately the manner originally proposed.

APPENDIX B

Progress Report No 2
(1 January 1978 - 31 March 1978)

Progress during this period has been characterized by (1) continuing implementation of previously designed software and (2) development of the overall design of the EUCLID translator. Our next major delivery of software is to be a translator for EUCLID that produces PDP-11 code. This translator should translate itself as its first major task. We plan to deliver the translator in July 1978, but this date could possibly slip.

The project is now perhaps three months behind where we had originally hoped to be. The major delay has been the continuing effort required to disambiguate the EUCLID language specifications. This effort is documented in a large notebook of ARPANET communications that we have had with the EUCLID committee. This effort was not included as a part of the original implementation project proposal. Despite this delay, no major difficulties are foreseen in completing the project.

In detail, the period's progress included:

1. Meeting with the EUCLID language design committee in January to iron out a number of language specification problems.
2. Production, testing and distribution of a parser for the full EUCLID language.
3. Design of a production I/O system for interfacing to UNIX.
4. Design and specification of streams for translator interpass communication.

5. Allocation of responsibilities to translator passes.

6. Design of disk-resident structures for symbol table and type table.

7. Design of mechanisms for supporting parameterized types.

8. Design of type and symbol table mechanisms for the translator.

9. Preparation of a number of project working papers to document the design of the translator.

These are overall designs which will evolve somewhat during detailed design and actual implementation.

APPENDIX C

ARPANET Communications Concerning EUCLID Language

SECTION 1: Clarification Requests

SUBJECT

CR#1 Semicolon rules

CR#2 When can "=" be used

CR#3 Uses of "with"

CR#4 Imported enum. types

CR#5 Access to Internal Module Variables

CR#6 Literal Tags for Variant Records

CR#7 "Const" vs. "Readonly" for imports

CR#8 "Const" vs. "Readonly" vs. "Var" for modules

CR#9 Allow "containing variable" for "import"

CR#10 End record vs. end indentifier

CR#11 Meaning of "opaque"

CR#12 Forcing import of "readonly" variables

CR#13 Meaning of first and last

CR#14 Finding ordinal of enum. value

CR#15 Functions that do not return values

CR#16 Exporting only some enum. values

CR#17 Readonly modules that import variables

CR#18 Holes in records

CR#19 Implicitly importing a collection

CR#20 (An official Hard Problem) Naming exported types

SECTION 1: Clarification Requests

SUBJECT

CR#21 Aliasing asserts for pointers

CR#22 Generality of returned types

CR#23 Preassertions in type declarations

CR#24 Type compatability rules

CR#25 "Any" as parameter of C.New

CR#26 Exporting enum. values

CR#27 Passing "size" to "allocate"

CR#28 Impossible assertions using components of exported types

CR#29 Subscripts in type names, legality assertions

CR#30 Components of formals, assertions

CR#31 Re: details pg. 31-32

CR#32 Pg. 32 line 5 and collections

CR#33 Compat., well-behaved for sets

CR#34 Assertions for missing case element

CR#35 Consistency and spelling rules

CR#36 Identifier after module

CR#37 Initialization affects compatibility rules?

CR#38 ItsType for M.D. record fields

CR#39 Consider ".." to be operator?

CR#40 Dummy Argument in EUCLID

CR#41 Impossible Assert for type actual range

CR#42 A modest proposal for Legality Assertions via"?"

CR#43 Order of destruction of array elements

SECTION 1: Clarification Requests

SUBJECT

CR#44 Assertion for function at call

CR#45 Assertion for value in return

CR#46 More on returning values

CR#47 Actual values for unknown in New

CR#48 Violation of static need-to-know

CR#49 Evading scope of converters

CR#50 Why export type with type?

CR#51 Position of invariant

CR#52 Use of half-defined collections

CR#53 Generality of type formals

CR#54 Order of case variants

CR#55 Dangling bindings

CR#55a Empty Subranges

CR#56 Record scopes, opening via dot

CR#57 Anomaly on legality of <<=

CR#58 Unspeakable assertion for tags

CR#59 Parameterized module types

SECTION 2: Interpretations

SUBJECT

Interp#46 Compat of built-in types

Interp#47 When initialization is (not) part of type

Interp#48 Use of variables in initialization

Interp#49 Order of initialization

Interp#50 Compat. of parameterized modules

Interp#50a Init. of record fields but not array cmpnts

Interp#51 Multiple use of tag assumed illegal

Interp#52 Parameter cmpts of types are not inherited

Interp#53 Variant incrementing of coll. counts

Interp#54 Range of variant tag is manifest

Interp#55 Init. of variant records

Interp#56 Manifest types for structured constants

Interp#57 Examples of record/module scopes

Interp#58 No Var. Recs. for struct. consts.

Interp#59 Standard components in assertions

Interp#60 Recursion only with explicitly importation

Interp#61 Abstraction function considered to be comment

Interp#62 Illegal e.g., pg. 31

Interp#63 Naming enum. values, scope rules

Interp#64 Literals, exported types, "Real" example

Interp#65 Error in "Real" example comparison?

Interp#66 Definition of manifest constant

Interp#67 Use of "unknown"

SECTION 2: Interpretations

SUBJECT

Interp#68 Dangling pointers

Interp#69 Error of $N for $$N

Interp#70 Confusion on scope inside "loop"

Interp#71 Wierd loop on pg. 53

Interp#72 Id. following "module"

Interp#73 Illegal type component e.g.

Interp#74 Missing "readonly" pg. 40

APPENDIX D

Minutes - Meeting
(6-7 January, 1978)

The numbering of issues in the report matches those in 78-3 and 78-4 from Wortman, with a few extra items appended to the end. I have put them in order of discussion, rather than in numeric order.

STRINGS

We propose changing the report to use the CSRG-designed string type in place of the current type. We will change the example to use them also, except that we may use the current string type in an example to illustrate how other string-like types can be defined.

VARIABLES THAT START WITH (

These are now illegal.

EXPORTED TYPES THAT IMPORT VARIABLES

If a (module) type, Inner, inside another module type, Outer, imports a VAR, then Inner cannot be exported from Outer.

MANIFESTNESS OF FORMALS

See RESTRICTIONS ON FORMAL PARAMETERS OF TYPES in this report.

DANGLING POINTERS

1. Add the optional attribute CHECKABLE to a collection type definition.
2. Add the standard component refCount to dynamic variables in CHECKABLE or COUNTED collections.

3. The appropriate legality assertion for C.Free(v) is v.refCount=1.

4. It is illegal to use C.Free in a checked scope unless C is CHECKABLE.

Notes: (a) A CHECKABLE collection pays the reference counting overhead in all scopes, whether CHECKED or UNCHECKED.

(b) The implementation for a CHECKABLE, uncounted collection, CUC implements a var p as a pointer to a two-component record (allocated from the system zone, not from CUC) with another level of pointer to the value as one component, and a reference count as the other.

RETURN VALUES

They will remain as currently defined in the Report.

RESTRICTIONS ON FORMAL PARAMETERS OF TYPES

The following rule, proposed by Butler Lampson, was accepted: "If you import a parameterized type, you must also import the identifiers used in its formal parameter list, exclusive of its formal parameter identifiers."

SYNTAX FOR DECLARING TYPE CONVERTERS

The syntax proposed by the implementation team was accepted; an identifier so-declared obeys normal scope rules.

COMPONENTS FOR EXPRESSIONS

The type of an INTEGER expression is INTEGER. The standard components first, last, and size are not defined for type INTEGER. A 'value' can only be a variable, a constant, or a function call (not a generalized expression).

PERVASIVE IMPLIES POTENTIALLY RECURSIVE

Toronto restriction accepted; i.e., PERVASIVE does not imply automatic importation of type/routine name into its own scope.

ORDER OF FINALIZATION

The order of initialization for an array A is in increased order from A.IndexType.first through A.IndexType.last; for a record, the components are initialized in the order (left-to-right) in which they were written in the record definition. For both initialization and finalization, a contained variable is initialized/finalized resp. by initializing/finalizing its component parts. Finalization is done for arrays and records in the reverse order of initialization. This is consistent with the order of finalization for the variables and constants in a given scope, since it is treated as a nested set of micro scopes, each beginning at a new declaration (thus, the finalization for the variables looks more like the inside-out of the rule than the rule of components at the same level).

COUNTED COLLECTIONS

Yes, lots of counting for assignment (I45); yes, variant record affects counting (I53), bindings are treated like pointer assignment; deallocating pointers into counted collections requires decrementing reference counts. There is a problem with module assignment ("m1 := m2" smashes m1 but no FINALLY will be done for it, and two copies of the original value of m2 will later be finalized), so assignment of modules with

FINALLY is prohibited (Jim H. remembers this being decided, others do not; I think that we should accept it as a decision anyway).

VARIANT RECORDS

itsTag standard component accepted. Tag field can be a literal or expression (use the OTHERWISE label to handle the possible infinity of leftover cases). Case labels cannot be non-manifest constants. Three proposals were presented for initializing the tag of a variant record ((b) was the one accepted):

(a) v: T(ANY :=red) {default specified when a variant record declared, not when the type is defined}

(b)
```
TYPE T(tag: Color) = RECORD
        CASE tag DEFAULT red OF {the "DEFAULT red"
                                 part is optional}
            red => . . .
            green => . . .
            . . .
            END CASE
```

(c) Do nothing

A new syntax was accepted for the discriminating case statement. Change the syntax on p. 51 of the report.

```
simpleCase ::= CASE expression caseTail
discriminatingCase ::= CASE object caseTail
caseTail ::= OF caseBody END CASE
```

If a variant record variable is declared with ANY or with a non-manifest tag, it must be discriminated to access the variant components. The labels in a variant record declaration are restricted to being identifiers, literals or ranges.

LEGALITY ASSERTION FOR FUNCTION RETURN VALUE

The return-value identifier is now mandatory.

LEGALITY ASSERTION AT POINT OF FUNCTION CALL

The legality assertion is speakable.

EXPORTING THE ABILITY TO POINT TO

Can export WITH .

MANIFESTNESS OF STRUCTURED CONSTANTS

Structured constants must be manifest.

WELL-BEHAVED RULE

As specified in message #48 (PARC msg number), 13 SEPT 77.

PROPERTY X

Still a hard problem; CHECKABLE collections may help.

SEPARATE COMPILATION

Report stands unchanged.

ITEMS ADDED DURING THE MEETING:

TRANSITIVE CLOSURE OF IMPORT LISTS

The following syntax was developed (with Mitchell's strong dissent) for the compiler to produce an annotated listing that would be acceptable compiler input later (and also to allow a list of import lists instead of just one). The identifier list after THUS would be supplied by the compiler in the listing and ignored by it if seen as input:

```
importClause ::= singleImportClause
                 {";" singleImportClause} | empty

singleImportClause ::= IMPORTS importList [THUS importList]

importList ::= "(" importItem {"," importItem} ")"
```

Rules about implicitly imported identifiers:

(1) An implicitly imported identifier cannot be redeclared in any scope into which it is implicitly imported.

(2) An implicitly imported identifier cannot be used in any scope in which it is so imported; if it is needed, it must be imported explicitly.

NEW VERSION OF THE REPORT

The Palo Alto-based contingent of the committee will produce a new version of the EUCLID report incorporating all known changes through January 7, 1978 by mid-March, 1978.

MESSAGES ABOUT EUCLID

We will publish a memo saying how we will coordinate messages and responses by the end of January.

PROOF RULES

No more changes will be made to the Acta Informatica paper before it is published.

ASSIGNMENT OF ARRAYS/RECORDS WITH VARIANT COMPONENTS

This will not be allowed, and the report will be changed to make this clear (especially p. 32).

THE TYPE-SAMENESS RULE

Most of the heat (and little of the light) in the meeting surrounded this discussion. It was finally voted on with the outcome in favor of the current rule, revised as indicated in various places (Votes: for Toronto rule: 2, for Report rule: 6, abstentions: 2). As penance, the committee will write a complete sameness rule with all the fixes and send it to the implementation team soon.

Jim Mitchell

P.S. The implementation team and I have already gone over these minutes together; this is the final, approved version.

APPENDIX E

EUCLID Working Paper Notebook

NUMBER	TITLE
1	On Legality Assertions in EUCLID
2	A Possible EUCLID Compiler Structure
3	Structure of the Scanner and Screener
4	Programming Conventions
5	A Syntax/Semantics Language
6	Small EUCLID
7	The Syntax of Small EUCLID and Small C
8	Screener Output Files
9	File Input/Output Routines
10	A Child's Guide to Imports and Exports
11	A User-Oriented Syntax of Full EUCLID
12	A Discussion of "A User-Oriented Syntax of Full EUCLID"
13	Format of the Syntax/Semantic Tables
14	A Run-Time Model for EUCLID
15	EUCLID Language-Defined Identifiers
16	Notes on EUCLID Compiler Structure
17	Procedure Linkage on the PDP-11 (SUE.8 Working Paper 9)
18	Constant Folding in Postfix Expressions
19	Syntax of the Parser Output for the Full EUCLID Translator
20	Input/Output in EUCLID

NUMBER	TITLE
21	Some Major Tasks for Jan 1 - July 1, 1978 EUCLID Implementation
22	The Sizer - A Part of the Conformance Checker
23	The Dot Interpreter - A Part of the Builder
24	The Constant Folder
25	Evaluating Literals and Folding Constants
26	Proposed EUCLID Translator Structure
27	Notes on the Structure of the EUCLID Translator
28	Value Descriptors
29	Type Table - Detailed Description
30	Symbol Table - Detailed Description
31	Variant Records and Discriminating Cases
32	Interface to Disk-Resident Tables
33	Set Operations in Small EUCLID
34	Compiling Parameterized Types

Distribution List for Technical Reports

ARPA, Attn: Program Management — 2 copies
1400 Wilson Blvd.
ARLINGTON, VA 22209

Dr. G.X. Amey — 1 copy
CRAD DST(SE)4
Department of National Defence
101 Colonel By Drive
OTTAWA, Canada K1A 0K2

Defense Documentation Center (DDC) — 12 copies
Cameron Station
ALEXANDRIA, VA 22314

Letters of Transmittal sent to:

Mr. Ken Laver
Science Procurement Branch, CCC
11C1 Place du Portage, Phase III
11 Laurier Street
HULL, Quebec K1A 0S5

Defense Contract Administration Service
Management Area, Ottawa
219 Laurier Avenue, West
6th Floor
OTTAWA, Canada K1A 0S5

Mr. Steve Walker
OSD-CCCI
The Pentagon
WASHINGTON, D.C. 20310