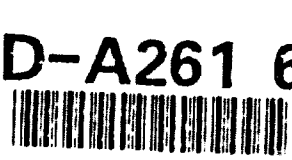

(2)

TOWARDS A FORMALISM FOR PROGRAM GENERATION 1992 - FINAL REPORT

Daniel E. Cooke

University of Texas El Paso

1992

AFOSR
Air Force Office of Scientific Research

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		FINAL/15 JUN 89 TO 29 DEC 92

4. TITLE AND SUBTITLE

TOWARDS A FORMALISM FOR A PROGRAM GENERATION (U)

5. FUNDING NUMBERS

2304/FS
F49620-89-C-0074

6. AUTHOR(S)

Professor Daniel E. Cooke

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Texas
Computer Science Department
El Paso, TX 79968-0518

8. PERFORMING ORGANIZATION REPORT NUMBER

AFOSR-TR- 93 0083

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFOSR/NM
110 DUNCAN AVE, SUTE B115
BOLLING AFB DC 20332-0001

10. SPONSORING MONITORING AGENCY REPORT NUMBER

F49620-89-C-0074

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION IS UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The following was accomplished over the period of the contract: (1) Studied the interaction between iterative and data structures; (2) Completed Denotational Semantics of BagL; (3) Initiated work on a logical semantic for BagL; (4) Initiated work on a Visual Interface for BagL; (5) Initiated work on semantic extensions to support software maintenance in BagL; (6) Initiated a revision of BagL semantics; and (7) Initiated work on a BagL interpreter. In the coming years the researchers hope to complete the BagL interpreter, the logical semantic, and establish the expressiveness of BagL. In the long term it is hoped to apply results of nonmonotonic logic research to BagL for the purpose of software evolution automation. They also hope to develop a visual interface based upon the formal language.

14. SUBJECT TERMS	15. NUMBER OF PAGES	16. PRICE CODE
	24	

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR(SAME AS REPORT)

TOWARDS A FORMALISM FOR PROGRAM GENERATION 1992 - FINAL REPORT

Prepared by
Daniel E. Cooke

Department of Computer Science
University of Texas El Paso
El Paso, Texas 79968-0518

December 1992

Contract F49620-89-C-0074

Prepared for

AFOSR
Air Force Office of Scientific Research
Mathematical and Information Sciences
Air Force Office of Scientific Research
Bolling Air Force Base
Washington, D.C. 20332 6448

REPORT DOCUMENTATION PAGE	1. REPORT NO.	2.	3. RECIPIENT'S ACCESSION NO.

4. Title and Subtitle	5. Report Date
Towards a Formalism for Program Generation 1992 - Final Report	December, 1992
	6.

7. Authors	8. Performing Org. Rept. No.
Daniel E. Cooke	

9. Performing Organization Name and Address	10. Project/Task/Work unit No.
Department of Computer Science University of Texas El Paso El Paso, Texas 79968-0518	11. Contract No. F49620-89-C-0074

12. Sponsoring Organization Name and Address	13. Type of Report & Period Covered
Air Force Office of Scientific Research Mathematical and Information Sciences Bolling AFB Washington, D.C. 20332 6448	Final Report 6/89 - 12/92
	14.

15. Supplementary Notes

16. Abstract (Limit: 200 words)

The report which follows is the result of the work accomplished during a three year contract with the AFOSR. We have accomplished the following:

1. Studied the interaction between iterative and data structures;
2. Completed Denotational Semantics of BagL;
3. Initiated work on a logical semantic for BagL;
4. Initiated work on a Visual Interface for BagL;
5. Initiated work on semantic extensions to support software maintenance in BagL;
6. Initiated a revision of BagL semantics; and
7. Initiated work on a BagL interpreter.

In the coming years we hope to complete the BagL interpreter, the logical semantic, and establish the expressiveness of BagL. In the long term we hope to apply results of nonmonotonic logic research to BagL for the purpose of software evolution automation. We also hope to develop a visual interface based upon the formal language.

17. Document Analysis a. Descriptors

b. Identifiers/Open-ended Terms

c. COSATI Field/Group

18. Availability Statement	19. Security Class (report)	21. No. of Pages
	20. Security Class (page)	22. Price

NOTICES

This final report is a presentation of the findings from research funded under F49620-89-C-0074.

ABSTRACT

The report which follows is the result of the work accomplished in the second year of a three year contract with the AFOSR. In the past year we have accomplished the following:

1. A definition of a computational expression in L0;
2. A definition of stratification as applied to requirement specifications;
3. A definition of a synthesis method for computational expressions;
4. A precise understanding of ambiguity, completeness, and consistency with respect to L0;
5. A definition of the property expression;
6. A refinement of L0;
7. A synthesis program for computational expressions; and
8. A goal for an abstraction level for L0.

We hope to complete the definition of L0 according to the outlined syntax given in section 2.4. We intend to develop formal semantics for the language. We request that the AFOSR consider an amendment to our current contract to allow us to continue our development of this language including proofs of correctness and proofs to show that the language does not limit the ability to solve problems (i.e., the proofs we obtained for L1 in Cooke90-1).

We believe that ultimately, we can build a workstation around the Sun Sparc Station2 which is based on the language L0.

TABLE OF CONTENTS

FOREWORD

This is the final report associated with AFOSR contract, F49620-89-C-0074.

PREFACE AND ACKNOWLEDGEMENTS

Research sponsored by the Air Force Office of Scientific Research (AFSC), under contract F49620-89-C-0074. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

SYMBOLS AND ABBREVIATIONS

UTEP - University of Texas El Paso

$\neg$ - Logical NOT
$\wedge$ - Logical AND
$\vee$ - Logical OR
$\rightarrow$ - Logical IMPLICATION
$\leftrightarrow$ - Logical EQUIVALENCE or BICONDITIONAL
τ - Logical TERM
Φ - Metasymbol to represent $\wedge$, $\vee$, or $\neg$.
Ψ - Well-formed formula.
ψ - Atomic formula.
$\forall$ - Universal Quantifier.
$\exists$ - Existential Quantifier.
Π - An arbitrary program.
Σ - An arbitrary program specification.
$\Rightarrow$ - "such that".
$\Leftarrow$ - "if".
$\leftarrow$ - "defined in terms of".

SUMMARY.

The goal of this funded effort was to study a representative third generation language, to ascertain which (if any) language control constructs could be abstracted out of the language in order to achieve a higher level abstraction for problem solving with computers. The study focused on whether the higher level language was inherently ambiguous.

In year 1, the study of abstracting out selective and iterative constructs was accomplished. It was determined that the elimination of the selective structure resulted in a high degree of ambiguity. Furthermore, it was determined that abstracting out selective structures actually hampered the ability to solve problems. This work appears in [Cooke89, Cooke90-1, Cooke90-2].

After a very thorough study of iterative structures (based upon the directions established in [Cam, Lis, Shaw]), it was discovered that given a Turing Computable LISP-like language, ambiguity occurs when iterative/recursive constructs are eliminated. The resulting ambiguity can be summarized in the following way. Given that one variable is to be used to produce a result in another variable, and no information is given as to whether the variables contain atomic or list values, one may be producing any of the following computations:

1. An atom from an atom (e.g., x := y+10);
2. An atom from a list (e.g., computing the sum or product of a list);
3. A list from an atom (e.g., computing Fibonacci sequence from two atomic seeds); or
4. A list from a list (e.g., producing a sorted list from an unsorted list).

The details concerning this study can be found in [Cooke90-3, Cooke91-1, Cooke91-2, Cooke91-3, Cooke91-4, Cooke92-1, Gates90, Gates92, and Cooke92-2].

In order to avoid the ambiguities revealed in the year one studies, the high level language, BagL was developed. This language has a single, basic data structure which can be *configured* into any imaginable structure. This basic data structure (or metastructure) is to be used to represent a display, report, database, response to a realtime event, etc. The language allows the specifier to describe the basis of membership in the structures (e.g., formulae to compute the elements or formulae to state the manner in which elements will be selected from some source of data, etc.). The language also allows the specifier to state the organization of the structure in terms of the ordering or partial ordering of elements and the composition of nested structures. The structures exist as persistent structures, wherein there is no conversion necessary between internal and external structures [Lamb].

Having a single "metastructure," at first thought, seems to be a step backwards. However, actual data structures to be built to solve a problem are less of a specification problem than they are an implementation problem. In other words, when **specifying** the solution to a problem one may claim to need a list, but one is unlikely to say that the list should be implemented as an array or linked list, etc.

The metastructure possesses the additional advantage that it is easier to infer the algorithms to process a single data structure than it is to infer the algorithms to process different types of data structures. Therefore, BagL does not require the specifier to state the algorithmic solution to the problem. In other words, the ability to specify control structures is effectively abstracted out of the language. By specifying only the "data structure" and not the algorithm, the specifier is indeed specifying (albeit in detail) the external behavior of the system.

The nature of such a language in terms of its target abstraction level is an extremely important decision. According to [Wirth] "One of the most crucial steps in the design of a language is the choice of abstraction upon which programs are to base." Wirth goes on to say that it is important to select abstract objects of a language from the same level. The single metastructure of BagL is the ordered multiset or ordered bag. A bag may consist of a singleton or it may consist of ordered bags of ordered bags.

BagL is object oriented in that the focus in specification is upon the object produced by the computer -- the focus is not on the program which produces the object. Furthermore, BagL may recommend a visual or even virtual reality interface to allow the specifier to "visualize" and "visually construct" the data structures. The introductory information concerning BagL appears in [Cooke92-3, Cooke92-4, Cooke93, Pedroza92]. BagL possesses the following characteristics:

(1). provides a coherent and "better" abstraction level for problem solving,
(2). is unambiguous (i.e., possesses formal semantics),
(3). is general purpose and extensible,
(4). is concise in terms of features and constructs,
(5). allows for the appropriate interaction between features and constructs,
(6). provides for a correct transformation from specification to an efficient executable form,
(7). provides facility to reason about intended programs, and
(8). provides support for software maintenance or evolution.

A logical semantic (separate from the program semantic) of BagL is being developed. The intention is to provide for a straightforward ability to reason about intended BagL programs. Furthermore, results from the study of nonmonotonic logics [Gel] are to be applied to BagL to facilitate maintenance of BagL software. The preliminary research in this direction appears in [Luqi, Ram91, and Ram92]. Finally, preliminary work on a concurrent semantic for BagL is being performed with initial work appearing in [Cooke92-5].

Currently, the formal semantics of BagL are being revised and a BagL interpreter is being written in Prolog. The interpreter and the application of nonmonotonic logic results to BagL go beyond the original goals of this project.

1. INTRODUCTION.

When solving a problem, a problem solver faces two major concerns - the problem to be solved and the technicalities involved in using the tools available to solve the problem. For example, when a mechanic removes a spark plug, he/she must focus on the removal of the spark plug *and* the technicalities involved in using a socket wrench as a tool to remove the plug. We call the skills used for the former, general problem solving skills and the skills employed in the latter, tool skills.

In order to use the socket wrench effectively, the mechanic may need an extension to be attached to the socket wrench upon which the appropriate socket is to be placed. Once the tool has been set up, the mechanic must use it properly in order to remove and not tighten the plug. After the tool set-up the mechanic shares his/her attention between using the tool correctly and solving the problem at hand.

Tool skills are necessary in order to overcome the complexities inherent in using the tool. Past efforts to improve the problem solving environment have involved abstracting out some focal point of tool complexity, so that the problem solver can better focus on the general problem solving and problem domain complexities. Therefore, each of the previous advancements in problem solving **languages** resulted from attempts to abstract out the focal point of complexity in the less sophisticated language.

Machine complexities (i.e., register usage, memory management, etc.) provided this focal point in early problem solving environments. In the development of FORTRAN, standard assembler programming practices were identified and abstracted out, resulting in a language which insulated the programmer from machine complexities. However, FORTRAN imposed no standard control structures. Thus, the control structure complexity provided the focal point for further language improvement.

In Algol and Pascal, standard ways to set up selective and iterative control structures were identified, virtually eliminating the need for the GO TO statement and the complexity which arises from its use. Once again, through generalization of a tool-based complexity, software engineers were elevated in their abstraction level, allowing for a focus on a more sophisticated class of complexity. Data-Control structure complexities have been the focus of much of the current SPM as suggested by [Ovi, Boehm, Curtis, Shneiderman].

In fact, large amounts of time are spent in the design of the interaction of data-control structures. Hence, in recent years much effort has gone into the development of areas such as Abstract Data Types, Data Flow Design methods, Data Encapsulation, etc. If it is possible to abstract out data-control structure interactions, a true next generation language may result.

Many very high level languages recently developed reflect the tendency of software engineers to identify features which are common to all of the software products they have developed within some well-defined problem domain. Unfortunately, an understanding of past successes only assures that if future

problems are more or less like past problems, they can probably be solved with the language. However, there is no guarantee of success.

When a new problem is encountered that cannot be solved by the domain specific language, there is a tendency to add features to the language so that it can solve the problem. Zave claims that extending the applicability of a language can easily damage its coherence [Zave]. Coherence has to do with whether or not a language maintains a consistent level of abstraction or representation.

When a language becomes incoherent, serious problems may arise. Consider a hypothetical version of a procedural language like Pascal which allows the programmer to *dip* into the assembler level of representation. If a change is made to a program in this language, the programmer must endeavor to understand the change at the Pascal and the assembler level. In other words, the programmer must understand the interaction of the machine code generated by the compiler and the assembler code he/she embedded in the source program.

Rather than beginning with a problem domain, the development of BagL began with attempts to abstract out the focal point of tool complexity inherent in the procedural programming languages. In other words, we began with an existing general purpose language, and attempted to eliminate the constructs which complicate problem solutions in that language (i.e., we have abstracted out control constructs).

2. METHODS, ASSUMPTIONS, AND PROCEDURES.

2.1 An Overview of the Current Version of BagL.

The formal Syntax and Semantics of BagL are based upon general loop constructs where an infinite amount of memory is available. Practically speaking, the majority of the memory of BagL will consist of external disk files. The basic notion of BagL is the ordered bag (or multiset). All objects (be they reports, screens, or databases) are ordered bags: from singleton bags to ordered bags of ordered bags.

The general form of a BagL program is Π = **(main,ƒ1,ƒ2,....,ƒn)** , where main and each ƒi is of the general form:

function_name = {**domain,range** | D1 & D2 &...& Dm}
where
1. **domain** and **range** are lists of bag variables and
2. each Di is of the general form: if **precondition** then **postcondition.**

A precondition consists of a formula wherein relations are connected by & (and), OR (or), and ~ (not). Relations are formed using the functors >, <, <=, >=, ≈, and ≠. These are the standard algebraic relations, except that they are redefined to operate with nonscalars (i.e., bags). The precise semantics of the relations are beyond the scope of this paper.

In BagL there exists an environment E associated with every program Π. The environment E(Π), consists of tuples which pair variable names (called Bag Variables, V's) and the actual values associated with the variables. The the contents of V's named in the domain list of the main function are "input," in that they are copied from the user environment to E(Π). When main completes execution, the contents of all range variables of the main function are copied out of E(Π) to the user environment. Thus results are output.

Similarly, when a function ƒi executes, it has an environment E(ƒi). The function calling ƒi copies the values of all domain variables to E(ƒi). When ƒi completes execution, the values of all range variables are copied out to the calling function's domain. Clearly, functions may be recursive. Consider the following general example:

E(ƒa)= { <x,((1),(2))>, <y,((3),(4),(5),(6))> }

Assume ƒa contains, in its postcondition, the relation :=(z,ƒb(x,y)) and

ƒb=((m,n),p| if true then :=(p,+(m,n)))

When :=(z,ƒb(x,y)) is executed, first ƒb's environment is established:

E(ƒb)= { <m,((1),(2))>, <n,((3),(4),(5),(6))> }.

After executing, ƒb's environment is

E(ƒb)= { <m,((1),(2))>, <n,((3),(4),(5),(6))>, <p,((4),(6),(6),(8))> }.

Upon return to ƒa, ƒb's environment is destroyed and ƒa's environment is:

E(ƒa)= { <x,((1),(2))>, <y,((3),(4),(5),(6))>, <z,((4),(6),(6),(8))> }

This notion of copy-in and copy-out is depicted in figure 1.

A postcondition formula consists of relations connected by a single, sequencing connector ";". The only possible relation in a postcondition formula is the assignment operation, ":=". Given :=(V,CB) in a function, ƒ, where V can be any possible variable and CB can be any possible bag, after execution of the assignment, <V,CB> ∈ E(ƒ) or if ƒ is main, <V,CB> ∈ E(Π).

For i = j-1, if the precondition formula of Di fails, the precondition of Dj is attempted next. When a precondition succeeds, the associated postcondition is executed. The syntax and semantics of the terms of both pre and postcondition relations are given precisely in the next sections.

2.2 Syntax of BagL Terms

What follows are the formal definitions of the syntax of BagL terms. The bags of BagL are made up of bags or atoms. Therefore, the definitions begin with the atom.

Def. ATOMS of BagL.

The atoms of BagL include real, integer, boolean, and string constants like those of extended versions of Pascal.

Def. SUBTERMS of BagL.

1. Letter λ_i is a subterm; and
2. if st is a subterm then pred(st) or succ(st) are subterms.

Def. UNARY OPERATIONS of BagL.

1. if st is a subterm and u is a unary operator such as sqrt, abs, etc., then υ(st) is a unary operation, o; and
2. if o is a unary operation and u is a unary operator, then u(o) is a unary operation.

Def. TERMS of BagL

1. () is a term;
2. if a is an atom then (a) is a term;
3. if t_1, t_2, ... t_n are terms and n≥2 then (t_1, t_2, ... , t_n) is a term (and is called a bag);
4. Variables x, y, z, X,Y, and Z are terms;
5. if **i** is a Greek letter or an integer constant, and t is a term, then t^i is a term;
6. if a_1 through a_n are integer atoms, then ((a_1),...,(a_i),_, (a_n)) is a term;
7. if a_1 through a_n are integer atoms, then ((a_1),...,(a_i), _F_ ,(a_n)) is a term where F is of the general form $\omega(o_1,o_2)$ or simply o_1 where ω can be an arbitrary binary operator and o_i are unary operations or integer atoms;
8. if t_1, t_2, ... t_n are terms and ƒ is a function symbol then ƒ[t_1, t_2, ... t_n] is a term;
9. if t_1, t_2, ... t_n, then ≥'[t_1, t_2, ... t_n] and ≤'[t_1, t_2, ... t_n] are terms;
10. Nothing else is a term.

The built-in function symbols include +, -, /, *, abs, sqrt, sqr, mod, size, etc. and special functions ≤' and ≥' for ordering bags. All other functions are user defined. All functions produce bags as results.

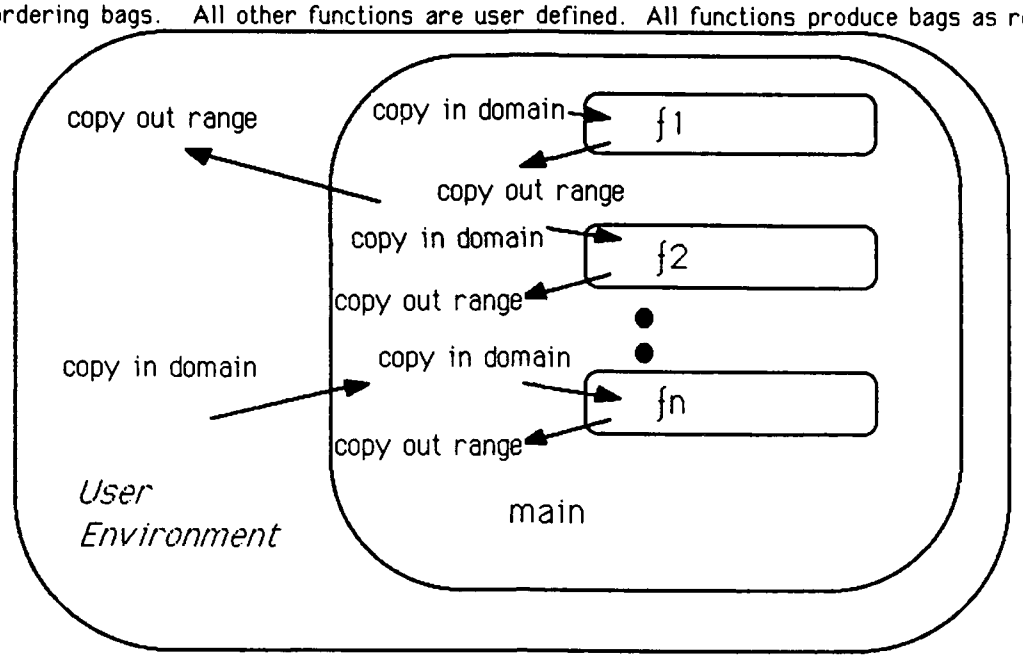

Figure 1.

2.3 Semantics of Terms

In the following, each item under the syntactic definition of the terms of BagL is reviewed. Item 1 provides for the empty bag. Item 2 states that an arbitrary atom (e.g., 5.2) is a term when placed in bag markers (5.2). This item provides for the construction of the singleton bag.

Item 3 provides for the composition of bags of bags. For example, (5.2), ((1),(2),(3)), and ('hello') are terms and so is ((5.2), ((1),(2),(3)), ('hello')).

Item 4 is concerned with the use of variables. Uppercase variables (called Bag variables) reference globally available bags. In practice, global bags are available as external files - thus eliminating the need to explicitly specify input and output. More formally, for a specification, there exists an execution time environment, **E**, of ordered pairs <V,CB> where all named bags exist. Named bags may be placed in the environment by the user (i.e., input) or by a specified function (i.e., computed or otherwise composed automatically). An actual bag, CB, is paired with its name, V, in the environment, E. The goal of a BagL specification is to discretely alter **E** to obtain some final $\mathbf{E_f}$ as indicated by the specification. An execution may be viewed as a sequence of $\mathbf{E_1}, \mathbf{E_2}, \ldots, \mathbf{E_f}$.

Lowercase variables are always quantified in a BagL specification and allow access to the components of an associated Bag variable.

Item 5 provides for a mechanism to obtain the position of some element of a bag. Associated with any element in CB is an ordinal position of the element. The ordinal position is obtainable through γ of V^{γ}.

The following semantic definition provides the formal meaning of item 6. Note, that **n**, in the following formulae, is an arbitrary value not necessarily known at the time a specification is given.

Rules 1-11 provide for the formal meaning of the built-in functions of BagL associated with item 8 of the syntactic definition of the term. The meaning of a user defined function is based upon the derived meaning of a user specification and is beyond the scope of this paper.

In the following semantic definitions, assume c_i is an arbitrary atomic constant; s_i is (c_i) or ($s_1, s_2, \ldots, s_n$); bag variables, or computed bags, T; ϕ stands for any binary operation; and σ stands for any unary operation. A bag consists of bag markers () enclosing a sequence of bags β_i, so that a bag is (βi). All operations begin according to rule 1, below (e.g., in general, ϕ[B1, B2,...]). Rule numbers appear the righthand margin. Finally, all unnecessary levels of parentheses are eliminated, i.e., ((x)) = (x).

Def. Simple Computed Bag of Integers.

Given $T=((a_1),\ldots,(a_i),\_,(a_n))$ The value of T is the bag $((c_1),\ldots,(c_i),\ldots,(c_n))$ where $c_1 = a_1$, $c_i = a_i$, $c_n = a_n$,etc. for an arbitrary element k, where $i \le k \le n$, $c_{k+1} = c_k + 1$ (1)

Example.

$((5),(8),\_,(12)) = ((5),(8),(9),(10),(11),(12))$ □

The next definition provides the meaning of item 7. The variable, λ_i always appears in the context of a list being generated. λ_i references the next element to be generated.

Def. Computed Bag.

Given, $T=((a_1),\ldots,(a_k),\_F\_,(a_n))$ The value of T is the bag $((c_1),\ldots,(c_k),\ldots,(c_n))$ where $c_1 = a_1, c_2 = a_2$, etc. for an arbitrary element i, where $k < i \le n$,

(1). subterm $pred(\lambda_i) = c_{i-1}$ and $pred(c_i)=c_{i-1}$

(2). subterm $succ(\lambda_i) = c_{i+1}$ and $succ(c_i)= c_{i+1}$

(3). $c_{i+1} = o(st_1)\ \omega\ o(st_2)$ where $F= \omega(o,o)$ and $c_{i+1} \le a_n$ or $c_{i+1} = o(st)$ (2)

Examples.

$((5),(8),\_+(pred(\lambda_i),3)\_,(21))=((5),(8),(11),(14),(17),(20))$

$((0),(1),\_+(pred(pred(\lambda_i)),pred(\lambda_i))\_,(45)) = ((0),(1),(1),(2),(3),(5),(8),(13),(21),(34))$

$((1.25),\_+(pred(\lambda_i),.01)\_,(1.30)) = ((1.25),(1.26),(1.27),(1.28),(1.29),(1.30))$ □

Definitions of Built in Functions.

$\phi[(B1, B2, \ldots, Bn)]$ (where $n>=2$) $= (\phi_{n-1}[\ldots(\phi_2[(\phi_1[(\beta 1^{x1}),(\beta 2^{x2})]),(\beta 3^{x3})]) \ldots,(\beta n^{xn})])$ (3)

where $m=max(|B1|, |B2|, \ldots, |Bn|)$ and x_i is the rational number $\frac{m}{|Bi|}$ and βi^{xi} indicates that the sequence inside Bi (i.e., βi) is repeated x_i times where the order of Bi is preserved. Consider the following example:

Example.

Assume $B1 = ((1),(2),(3),(4),(5))$, $B2 = ((10),(20),(30))$, $B3 = ((1),(2))$ and +(B1,B2,B3) is to be evaluated:

$m = max(|B1|, |B2|, |B3|) = max(5, 3, 2) = 5$,

x_1 is $\frac{5}{5}$ or 1 $\quad\quad \beta 1=(1),(2),(3),(4),(5)$

x_2 is $\frac{5}{3}$ $\quad\quad \beta 2=(10),(20),(30)$

x_3 is $\frac{5}{2}$ $\quad\quad \beta 3=(1),(2)$

Therefore B1 is repeated one time, the sequence of B2 is repeated $1\frac{2}{3}$ times, and the sequence of B3 is to be repeated $2\frac{1}{2}$ times resulting in the operation:

$(+[+[((1),(2),(3),(4),(5)),((10),(20),(30),(10),(20))],((1),(2),(1),(2),(1))]$.

Rule (3) decomposes an operation to a series of nested applications of the operator to no more than two bags. The sequential repetition of objects in smaller bags based upon the cardinality of the largest bag preserves the associative and commutative properties of addition and multiplication. □

σ[(B1, B2, ..., Bn)] = (σB1, σB2, ..., σBn) (4)

Rule (4) indicates how the unary operator is applied to a bag.

Example.
sqrt[((9),(16),(25))] = ((3), (4), (5)) □

size[(B1, B2, ..., Bn)] = (n) (5)

For rules 3, 4, and 5: If Bi is a variable, Vi, then Vi is replaced by constant bag CBi where <Vi,CBi> ∈ E. If Bi is a computed bag, Ti, then Ti is replaced by constant bag CBi according to the appropriate rule (1 or 2) above.

Rule 6 is the most primitive rule to apply an arithmetic operator to two atoms:

$\phi[(c_1), (c_2)] = (c_1 \phi c_2)$ (6)

Example. *[(4),(5)] = (20) □

Given ϕ[B1,B2], rule 7 applies when |Bi| > 1. *Effectively, rule 7 distributes an operation.*

$\phi[((s_1),(s_2),...,(s_n)),((s_{n+1}),(s_{n+2}),..., (s_{2n}))] = \phi[(s_1),(s_{n+1})], \phi[(s_2),(s_{n+2})] ,..., \phi[(s_n),(s_{2n})]$ (7)

Example.
+[((1),(2),(3)),((4),(5),(6))] =
+[(1),(4)], +[(2),(5)], +[(3),(6)] □

Example.
+[((((1),(2)), (3)), ((2),(4)))] =
+[((1),(2)), (2)], +[(3),(4)] □

Rule 8 applies to nested operations.

$\phi[(\alpha_1 \phi'[\beta_1]\beta_2)] = \phi[(\alpha_1\alpha_2\beta_2)]$ where $\alpha_2 = \phi'[\beta_1]$ (8)

where β_2 and/or α_1 are sequences which may be empty.

Example. +[((4),(3)), **-[(4),(3)]**] apply rule 15 then rule 2 = +[((4),(3)),(1)] apply rule 7, etc. □

Rules 9-11 apply when ϕ is $\leq$ or $\geq$ and reference item 9 of the syntactic definition of the term. Assume ϕ' is either $\leq'$ or $\geq'$.

$\phi'[(a_1),...,(a_n)] = ((c_1),(c_2),...,(c_k),...,(c_n))$ (9)

such that for each c_k there is some a_i and for every a_i there is some c_k and $(c_1 \blacklozenge c_2 \blacklozenge \ldots \blacklozenge c_k \blacklozenge \ldots \blacklozenge c_n)$ holds.

Example.
<='[((3),(4),(5),(9),(8),(1))]=((1),(3),(4),(5),(8),(9))

If B=(B1,...,Bn) and for each Bi, |Bi| = m and no Bi has a component that is a nonsingleton bag, then

$$\blacklozenge^*[B] = (\blacklozenge^*((c_{1,1}),\ldots,(c_{1,n})),\ldots,\blacklozenge^*((c_{m,1}),\ldots,(c_{m,n})) \quad (10)$$

$$\blacklozenge^*[B,Bi] = (((c_{1,1}),\ldots,(c_{1,n})),\ldots,((c_{m,1}),\ldots,(c_{m,n}))) \text{ such that } c_{1,i} \blacklozenge c_{2,i} \blacklozenge \ldots \blacklozenge c_{k,i} \blacklozenge \ldots \blacklozenge c_{m,i} \quad (11)$$

Example. Assume edb = (id,sal,name) where id=(((4),(8),(9),(2),(1))),
sal=(((40000),(36000),(23000),(56000),(100000))),name=(((chico),(groucho),(zeppo),(harpo),(garbo)))

<='[edb,id] = (((1),(100000),(garbo)), ((2),(56000),(harpo)),((4),(40000),(chico)),
((8),(36000),(groucho)),((9),(23000),(zeppo))) □

All of the semantics presented here have been implemented in a BagL interpreter written in Prolog. All examples have actually executed under the interpreter.

2.4 Complete Examples

In this section, the full vision of BagL is made evident through examples. Consider the definition to compute n!. Assume N is a singleton bag containing an integer.

ʃact = {N,NFACT | if >(N,(0)) then :=(NFACT, *[((1),_,(N))]) &if =(N,(0)) then :=(NFACT,(1)) }

Consider also the specification to square a matrix. Assume **X** is a two dimensional bag **X=(c1,c2,...,cn)** wherein for i=1..n, |ci|=n:

sq = {X,X' | (x\X)(if =(x,$X^{i,j}$) then :=($X^{i,j'}$, +[*[$X^{i,*}$,$X^{*,j}$]]))}

in which =(x,$X^{i,j}$) is the precondition and :=($X^{i,j'}$, +[*[$X^{i,*}$,$X^{*,j}$]]) is the postcondition. The superscripts in the formula are actually subscripts of the referenced bag, X. The precondition requires that for any element **x**, **x** is of some row **i** and column **j** of bag **X**. Notice that the actual values of a Bag's subscript are obtained in the precondition, where the specifier states, that for any **x**, **x** occupies position **i,j** of the bag **X**.

The postcondition states that the (i,j)th element of the bag **X'** is to contain the sum of the products of corresponding elements of row **i** and column **j** of bag **X**. The asterisk (*) as a subscript indicates all possible values of the subscript. Hence, the reference $X^{i,*}$, refers to row i and the reference $X^{*,j}$, refers to column j. If row i contains elements ((1),(2),(3)) and column j contains elements ((3),(2),(4)) the result of *[$X^{i,*}$,$X^{*,j}$] is the bag, ((3),(4),(12)). The result of +[(3),(4),(12)] is the singleton bag, (19). This would be the result assigned to $X^{i,j'}$.

Finally, assume there exists a bag of salaries, SAL=((23000),(45000),(55000)), and a bag of employees, EMP=((larry), (curley), (moe)). Assume Larry's salary is $23000, Curley's is $45000, etc. Therefore, the user establishes (as input) the initial execution time environment:

E_0 = {<EMP, ((larry), (curley), (moe))>, <SAL, ((23000),(45000),(55000)) > }

To create an employee database, the following specification is executed:

ʃdb={(EMP,SAL),EMPDB|if true then :=(EMPDB,(EMP,SAL)) }

The function ∫db demonstrates that BagL is used for schema processing on a database. The result of this specification is:

E_1 = {<EMPDB, (EMP,SAL)>,<EMP, ((larry), (curley), (moe))>, <SAL, ((23000),(45000),(55000)) > }

The remaining functions demonstrate that BagL also serves as a database manipulation and computational language (i.e., in addition to schema processing, BagL also plays the role of the query and host language). Suppose it is desirable to give all employees a ten percent pay increase, but first it is desirable to know how much money must be added to the budget to absorb the pay increase:

extra={EMPDB,REQUIRED | if=(EMPDB,(EMP,SAL)) then :=(REQUIRED,+[*[(0.1),SAL]]) }

The result of this specification is that <REQUIRED, (12300) > is added to the execution time environment above.

E_2 = {<REQUIRED, (12300) ><EMPDB, (EMP,SAL)>,<EMP, ((larry), (curley), (moe))>
<SAL, ((23000),(45000),(55000)) > }

Perhaps it is also wise to know what the entire salary budget should be for the pay raise:

sbudg = {EMPDB,SALBUD | if =(EMPDB,(EMP,SAL)) then :=(SALBUD,+[*[(1.1),SAL]]) }

The result of this specification is that <SALBUD, (135300) > is added to the execution time environment.

E_3 = {<SALBUD, (135300) ><REQUIRED, (12300) >,<EMPDB,(EMP,SAL)>,
<EMP,((larry),(curley), (moe))><SAL, ((23000),(45000),(55000)) >}

After careful consideration, the raise is given:

raise={EMPDB,EMPDB'| if =(EMPDB,(EMP,SAL)) then :=(SAL',*[(1.1),SAL]; :=(EMPDB',(EMP,SAL')) }.

After execution of this final specification, the execution time environment (which is available to the user externally) is:

E_f = {<SALBUD,(135300) ><REQUIRED, (12300) >,<EMPDB, (EMP,SAL)>,<EMP, ((larry), (curley), (moe))>,
<SAL, ((25300),(49500),(60500)) >}

These multisets are available to the user through screen views and reports. Thus no conversion from an internal to external data structure is necessary (i.e., BagL is a persistent language).

3. RESULTS AND DISCUSSION.

In this section we will review the results which have been obtained. Please see appendix A for the "Statement of Work" which appeared in our original proposal.

3.1 PROJECT PERSONNEL.

The results obtained are based on the work of a total of six people. Dr. Michael Gelfond, Professor of Computer Science at UTEP, received one month's salary each year for the technical assistance he lends to the project. Dr. Daniel Cooke receives one month's salary each year as project director and principal researcher. Ms. Ann Gates and Mr. Bassam Chokr both received their twelve month salaries from the project. Ms. Gates is a research associate and Mr. Chokr is a research assistant.

Mr. Chokr is working towards a Masters Degree in Computer Science and Ms. Gates is working towards a Ph.D. The research component of their respective degrees is to be based on the research they conduct for the project. Previously, Mr. Rito Delgado and Mr. Miguel Pedroza were funded on this project. Both Mr. Delgado and Mr Pedroza have completed Masters Degrees as has Ms. Gates. Mr Delgado works for Research, Analysis, and Maintenance in El Paso. Mr. Pedroza is employed at IBM, San Jose. Ms. Gates is a Ph.D. student at New Mexico State University where Prof. Cooke serves as her supervising professor.

3.2 RESULTS.

We have accomplished the following activities:

1. Study of interaction between iterative construct and nonscalar data structures;
2. Completed Denotational Semantics of BagL;
3. Initiated work on a logical semantic for BagL;
4. Initiated work on a Visual Interface for BagL;
5. Initiated work on semantic extensions to support software maintenance in BagL; and
6. Initiated work on a BagL interpreter.

It is felt that BagL may serve as an excellent language for quickly developing programs to process Telemetry Data for Satellite Groundstations. Mathematicians believe it will also serve as an excellent language for conducting experiments in Interval Mathematics. Further evidence of the success of our theoretical work can be observed in our publications. The following is a list of papers/books based on the AFOSR project at UTEP.

Books

The Impact of Computer Aided Software Engineering on Software Processes. World Scientific Publishers, Ltd. Contributors: Raymond Yeh, Peter Ng, Luqi, Joseph Urban, Ron Norman, W.D. Hurley, John Baker, Patrick Bobbie, WT Tsai, Greg Boone, Nick Bourbakis, etc. In press.

Software Automation. Contributors: CV Ramamoorthy, Raymond Yeh, Peter Ng, Luqi, Berzins, Joseph Urban, Murat Tanik, etc. In progress.

Journal Papers

D.E. Cooke, "Towards a Formalism To Produce a Programmer Assistant CASE Tool," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 2 No. 3, September, 1990, pp. 320-326.

D.E. Cooke and A. Gates, "On the Development of a Method to Synthesize Programs from Requirement Specifications," *International Journal on Software Engineering and Knowledge Engineering*, Vol 1 No 1, (March, 1991) pp. 21-38.

Daniel E. Cooke, "An Issue of the Next Generation of Problem Solving Environments," *Journal of Systems Integration*, Vol 1(2), (February, 1992) pp. 39-52.

C.V. Ramamoorthy, Daniel E. Cooke, and Chitta Baral, "Maintaining the Truth of Specifications in Evolutionary Software," to appear *International Journal of TAI.*.

Refereed Conferences/Proceedings

D.E. Cooke, "Proving Properties of Software Design Methods," *Proceedings of the First International Conference on Software Engineering and Knowledge Engineering*, June, 1989, pp.9-12.

A. Gates and D. Cooke, "An Introduction to the Recognition of Iterative Structures by a CASE Tool," *Proceedings of the Second International Conference on Software Engineering and Knowledge Engineering*, Skokie, Illinois, June, 1990 pp.201-208.

Daniel E. Cooke and Ann Gates, "On the Application of Stratification to Requirement Specifications," *Proceedings of the Second International IEEE Conference on Tools for Artificial Intelligence*, November, 1990, pp. 760-766.

Daniel E. Cooke, "Methods of Program Generation for Engineering Applications," *Proceedings of ASME Energy-sources Technology Conference* , Houston, Texas (January, 1991), pp. 15 - 20.

Daniel E. Cooke, Miguel Pedroza, and Ann Gates, "Interaction Of Data Structures and Primitive Operations of Language LO," *Proceedings of the Third International Conference on Software Engineering and Knowledge Engineering*, June 1991, pp. 78-83.

C.V. Ramamoorthy and Daniel E. Cooke, "The Correspondence Between Methods of Artificial Intelligence and the Production and Maintenance of Evolutionary Software," *Proceedings of the Third International IEEE Conference on Tools for Artificial Intelligence*, November, 1991, pp. 114-118.

Ann Gates and Daniel E. Cooke, "On a Fundamental Relationship Between Software Reuse and Software Synthesis," *Proceedings of Hawaii International Conference on System Sciences Vol. II*, Kauia, Hawaii (January, 1992) pp. 539-548.

Mike Pedroza and Daniel Cooke, "The Informal Semantics of BagL," *PD-Vol. 43, Computer Applications and Design Abstraction ASME 1992*, Houston, Texas (January, 1992) pp. 29 - 32.

Luqi and D. Cooke, "The Management of Uncertainty in Software Development," *IEEE COMPSAC '92*, Chicago, IL, pp. 381-386.

Daniel E. Cooke, "Issues Surrounding Specification Languages For Software Automation," *Proceedings of IEEE Fifth International Workshop on Computer Aided Software Engineering*, July 6-10, 1992, Montreal, Canada, pp. 120-123.

Daniel E. Cooke and Aida Gutierrez, "An Introduction to BagL," *IEEE Fourth International Conference on Software Engineering and Knowledge Engineering*, Capri, Italy, pp. 479-486.

Daniel E. Cooke, "Logical Development of a Petri Net Deadlock Analysis Program," *Proceedings of the Fourth International IEEE Conference on Tools for Artificial Intelligence*, November, 1992, pp. 230-233.

Daniel E. Cooke, "Arithmetic Over Multisets Leading to a High Level Language," *Proceedings Computers in Engineering Symposium 1993*, Houston, Texas (January, 1993) to appear.

Panel Papers

D. Cooke, T. Escamilla, and M. Gibson, "The Correspondence Between Methods of Artificial Intelligence and the Production and Maintenance of Evolutionary Software," *Proceedings of the Third International Conference on Software Engineering and Knowledge Engineering*, June 1991, pp. 114-115.

B. Blum, D. Cooke, X. Li, N. Minsky, and R. Semmel, "The Best Approach to Knowledge Representation for Software Engineering," *Proceedings of the Third International Conference on Software Engineering and Knowledge Engineering*, June 1991, pp. 166-167.

D. Cooke, M. Feather, S. Fickas, N. Minsky, P. Selfridge, D. Smith, and J.P. Tsai, "Is AI the Solution for Software Engineering?," *Proceedings of the Third International IEEE Conference on Tools for Artificial Intelligence*, November, 1991, pp. 10-12.

Valdis Berzins, Daniel E. Cooke, Luqi , Peter Ng, C.V. Ramamoorthy, Murat Tanik, Joe Urban, and Raymond Yeh, "Workshop on Software Automation," *IEEE Systems Integration Conference, Proceedings of 1992 IEEE International Conference on Systems Integration*, Morristown, NJ, (June 15-19, 1992) pp. 720-722.

Further evidence of our success is the following list of invitations Dr. Cooke has received during the three year period:

Program Committees

Second International Conference on Software Engineering and Knowledge Engineering (S.K. Chang, Conference Chair).

Third International Conference on Software Engineering and Knowledge Engineering (S.K. Chang, Conference Chair).

Fourth International Conference on Software Engineering and Knowledge Engineering (S.K. Chang, Conference Chair).

Second IEEE Systems Integration Conference (Raymond Yeh and Peter Ng, Conference Co-Chair).

Third IEEE Systems Integration Conference (Raymond Yeh and Peter Ng, Conference Co-Chair).

European Joint Conference on Engineering Systems Design and Analysis (A. Ertas and T. Derbentli, Conference Co-chair).

IEEE TAI '91 (Benjamin Wah Conference Chair).

IEEE TAI '92 (Stephanou Conference Chair).

Vice Program Chair *IEEE TAI '92*.

Symposium Chair *ASME Computer Applications and Design Abstraction '93*.

Chair of Workshop on Software Automation for Systems Integration Conference 1992.

Chair of Workshop on Software Automation for Software Engineering and Knowledge Engineering Conference, 1993.

Journal Activities

Associate Editor of the *Journal for Software Engineering and Knowledge Engineering*.

Book Review Editor of the *Journal for Software Engineering and Knowledge Engineering*.

Editing a special issue of the *Journal for Software Engineering and Knowledge Engineering* for June, 1991 Publication.

Reviewer for *IEEE Computer, IEEE Software, IEEE Transactions on Knowledge and Data Engineering, Journal of Systems Integration, Journal of Systems and Software, and Journal of Tools For Artificial Intelligence*.

Invited Talks

"Proving Properties of a CASE Tool," Texas A&M University, Hosted by Dr. John Leggett, April 16, 1990.

"Issues in Computer Aided Software Engineering," University of California, Berkeley, Hosted by Professor C.V. Ramamoorthy, April 30, 1990.

"Issues in CASE Technology Transfer," IEEE CASE '90 Fourth International Workshop, Irvine, California, December 6, 1990.

"The Development of a Requirement Specification Language," Texas A&M University, Hosted by Dr. John Leggett, February 6, 1991.

"Logic and Software Engineering," Panelist, Third International Conference on Software Engineering and Knowledge Engineering, June 1991.

"Ambiguity in Software Engineering," Panelist, Third International Conference on Software Engineering and Knowledge Engineering, June 1991.

"Expert Systems in Program Verification," Space Grant Consortium, Austin, Texas, June 18, 1991

"An Overview of CASE," Naval Postgraduate School, Hosted by Dr. Luqi, July-August, 1991.

"Program Synthesis: Generalizations for the Purpose of Program Synthesis," Naval Postgraduate School, Hosted by Dr. Luqi, July-August, 1991.

"Program Synthesis: Languages L2 and DecSpec Purpose of Program Synthesis," Naval Postgraduate School, Hosted by Dr. Luqi, July-August, 1991.

"Program Synthesis: Ambiguity Issues," Naval Postgraduate School, Hosted by Dr. Luqi, July-August, 1991.

"Is Artificial Intelligence the Answer for Software Engineering?" Panelist, Third IEEE Tools for Artificial Intelligence Conference, San Jose, CA, November, 1991.

"Software Automation Issues," 1992 IEEE International Conference on Systems Integration, Morristown, New Jersey, June 15, 1992.

"Novel Approaches to Systems Integration," 1992 IEEE International Conference on Systems Integration, Morristown, New Jersey, June 16, 1992.

"An Introduction to BagL," Naval Postgraduate School, Hosted by Dr. Luqi, August, 1992.

"An Application of CAPS to Support Software Maintenance," Naval Postgraduate School, Hosted by Dr. Luqi, August, 1992.

"Formal Methods in CASE," *IEEE CASE Fifth International Workshop,* Montreal, Canada, June 8, 1992.

"Software Engineering Support for Program Generation and Maintenance," Research Analysis and Maintenance, Hosted by Ray Day, August, 1992.

"Knowledge Acquisition and Process Acquisition," *IEEE Tools With Artificial Intelligence '92,* Arlington, Virginia, November, 1992.

With respect to the statement of work, we have completed the study leading to the unambiguous specification language as evidenced by the formal syntax and semantics of BagL. We are making excellent progress towards a BagL interpreter being written in Prolog.

4. CONCLUSIONS.

Currently, we can conclude that we have indeed realized the goal of developing an unambiguous executable specification language as evidenced by the denotational semantics of BagL. Furthermore, we have established the significance of the application of results of the study of nonmonotonic logic to specification languages for the purpose of enhancing the ability to automatically maintain software. Finally, it seems clear that BagL recommends a visual interface. Since the specifier is strictly defining a single structure, the bag, varieties of structural icons are unnecessary. However, a traditional visual interface (via a CRT screen) is not realistic in the long term in that the screen limits the size and dimensions of bags that a person may wish to specify. Therefore, a virtual reality interface may provide the best human interface for the formal language, BagL.

5. RECOMMENDATIONS.

The work on BagL is far from complete. There are no doubt further changes that will be required of the formal language. We propose to do the following work:

(1). Complete a BagL interpreter based upon the current syntax and semantics;
(2). Enhance the semantics of BagL to abstract out quantifier information;
(3). Show expressiveness (via LISP approach);
(4). Extend semantics of BagL to provide facilities for realtime specification and concurrency;
(5). Establish a logical semantic for BagL to provide for constraints;
(6). Provide automatic facilities to help maintain BagL software; and
(7). Develop a prototype BagL environment including a visual interface.

Clearly, we are recommending a large scale effort in the long term. We realize our limitations and do not recommend that we do all of this work in isolation based upon a single proposed effort. We intend to maintain our contacts and work with other universities (NPS, Berkeley, etc.) over the next several years. Furthermore, Cooke is now a member of the graduate faculty at NMSU where he can serve as supervising professor on Ph.D. dissertations.

U.T. El Paso (UTEP) provides an ideal setting for the recommended effort. The majority of C.S. faculty at UTEP are working in efforts directly related to the proposed work. Michael Gelfond and Chitta Baral work in the semantics of logic programming and in nonmonotonic logic. Gelfond and Lifschitz recently established the stable model semantics of logic programming [Gel]. Alexander Rabinovich joined the UTEP C.S. faculty in September, 1992. Rabinovich is well known for his contributions to the semantics of concurrency. The results of all the C.S. faculty are shared in weekly faculty seminars held at UTEP.

In addition to the deliverable interpreter, progress and success of this project will be measurable in terms of the number of refereed publications based upon the proposed research. In the research previously funded by AFOSR, 17 papers have already appeared in print. We believe that we should be able to publish minimally four papers for each year of funded activity.

REFERENCES.

[Berzins] Berzins, V., and Luqi, Software Engineering with Abstractions, Reading, Mass., Addison-Wesley Publishing Company, 1991.

[Boehm] B.W. Boehm, "Software Life Cycle Factors," *Handbook of Software Engineering*, Vick and Ramamoorthy, eds. Van Nostrand Reinhold Electrical/Computer Science and Engineering Series, New York, 1984, pp. 494-518.

[Cam] R. D. Cameron, "Efficient High-Level Iteration with Accumulation," *ACM Transactions on Programming Languages and Systems* 11,2 (April 1989), 194-211.

[Cooke89] D.E. Cooke, "Proving Properties of Software Design Methods," *Proceedings of the First International Conference on Software Engineering and Knowledge Engineering*, June, 1989, pp.9-12.

[Cooke90-1] "Towards a Formalism for Program Generation," Daniel E. Cooke, for the Air Force Office of Scientific Research, #F49620-89-C-0074, July, 1990.

[Cooke90-2] D.E. Cooke, "Towards a Formalism To Produce a Programmer Assistant CASE Tool," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 2 No. 3, September, 1990, pp. 320-326.

[Cooke90-3] Daniel E. Cooke and Ann Gates, "On the Application of Stratification to Requirement Specifications," *Proceedings of the Second International IEEE Conference on Tools for Artificial Intelligence*, November, 1990, pp. 760-766.

[Cooke91-1] Daniel E. Cooke, "Methods of Program Generation for Engineering Applications," *Proceedings of ASME Energy-sources Technology Conference* , Houston, Texas (January, 1991), pp. 15-20.

[Cooke91-2] D.E. Cooke and A. Gates, "On the Development of a Method to Synthesize Programs from Requirement Specifications," *International Journal on Software Engineering and Knowledge Engineering*, Vol 1 No 1, (March, 1991) pp. 21-38.

[Cooke91-3] Daniel E. Cooke, Miguel Pedroza, and Ann Gates, "Interaction Of Data Structures and Primitive Operations of Language L0," *Proceedings of the Third International Conference on Software Engineering and Knowledge Engineering*, June 1991, pp. 78-83.

[Cooke91-4] "Towards a Formalism for Program Generation," Daniel E. Cooke, for the Air Force Office of Scientific Research, #F49620-89-C-0074, July, 1991.

[Cooke92-1] Daniel E. Cooke, "An Issue of the Next Generation of Problem Solving Environments," *Journal of Systems Integration*, Vol 1(2), (February, 1992) pp. 39-52.

[Cooke92-2] Daniel E. Cooke, "Issues Surrounding Specification Languages For Software Automation," *Proceedings of IEEE Fifth International Workshop on Computer Aided Software Engineering*, July 6-10, 1992, Montreal, Canada, pp. 120-123.

[Cooke92-3] Daniel E. Cooke and Aida Gutierrez, "An Introduction to BagL," *IEEE Fourth International Conference on Software Engineering and Knowledge Engineering*, Capri, Italy, pp. 479-486.

[Cooke92-4] "Towards a Formalism for Program Generation," Daniel E. Cooke, for the Air Force Office of Scientific Research, #F49620-89-C-0074, July, 1992.

[Cooke92-5] Daniel E. Cooke, "Logical Development of a Petri Net Deadlock Analysis Program," *Proceedings of the Fourth International IEEE Conference on Tools for Artificial Intelligence*, November, 1992, pp. 230-233.

[Cooke93] Daniel E. Cooke, "Arithmetic Over Multisets Leading to a High Level Language," *Proceedings Computers in Engineering Symposium 1993*, Houston, Texas (January, 1993) to appear.

[Curtis] Bill Curtis et. al. "Measuring the Psychological Complexity of Software Maintenance Tasks with the Halstead and McCabe Metrics," *IEEE Trans. Soft. Eng.* Vol. SE-5 Number 2, March, 1979, pp.95-104.

[Dji] E. Djikstra, "Guarded Commands, Nondeterminancy and the Formal Derivation of Programs," *Communications of the ACM*, Vol. 18 No. 8, pp.453-457, August, 1975.

[Gates90] A. Gates and D. Cooke, "An Introduction to the Recognition of Iterative Structures by a CASE Tool," *Proceedings of the Second International Conference on Software Engineering and Knowledge Engineering*, Skokie, Illinois, June, 1990 pp.201-208.

[Gates92] Ann Gates and Daniel E. Cooke, "On a Fundamental Relationship Between Software Reuse and Software Synthesis," *Proceedings of Hawaii International Conference on System Sciences Vol. II*, Kauia, Hawaii (January, 1992) pp. 539-548.

[Gel] Gelfond, M. and Lifschitz, V, "The Stable Model Semantics for Logic Programming," In R.A. Kowalski and K.A. Bowen, editors, *Proc. 5th International Conference and Symposium on Logic Programming*, pages 1070-1080, Seattle, Washington, August 15-19, 1988.

[Lamb] Charles Lamb et.al., "The ObjectStore Database System," *CACM*, Vol. 34, No. 10, October, 1991, pp. 50-63.

[Lis] B. Liskov, A. Snyder, R. Atkinson and C. Schaffert, "Abstraction mechanisms in CLU," *ACM* 20,8 (August 1977), 564-576.

[Luqi] Luqi and D. Cooke, "The Management of Uncertainty in Software Development," *IEEE COMPSAC '92*, Chicago, IL, pp. 381-386.

[Ovi] Oviedo, E.I., "Control flow, data flow, and program complexity," *Proc. IEEE Computer Software and Applications* (Nov. 1980), pp.146-152.

[Pedroza92] Mike Pedroza and Daniel Cooke, "The Informal Semantics of BagL," *PD-Vol. 43, Computer Applications and Design Abstraction ASME 1992*, Houston, Texas (January, 1992) pp. 29-32.

[Ram91] C.V. Ramamoorthy and Daniel E. Cooke, "The Correspondence Between Methods of Artificial Intelligence and the Production and Maintenance of Evolutionary Software," *Proceedings of the Third International IEEE Conference on Tools for Artificial Intelligence*, November, 1991, pp. 114-118.

[Ram92] C.V. Ramamoorthy, Daniel E. Cooke, and Chitta Baral, "Maintaining the Truth of Specifications in Evolutionary Software," to appear *International Journal of TAI.*.

[Shaw] M. Shaw, W. A. Wulf and R. L. London, "Abstraction and Verification in Alphard: Iteration and Generators," in *Alphard: Form and Content*, edited by M. Shaw, 73-110, New York: Springer & Verlag, 1981.

[Shneiderman] Ben Shneiderman, *Software Psychology Human Factors in Computer and Information Systems*, Winthrop Publishers, Cambridge, Massachusetts, 1980.

[Wirth] Wirth, N., "On the Design of Programming Languages", in Programming Languages: A Grand Tour, Edited by E. Horowitz, PP. 23-30, Computer Science Press, Rockville, MD, 1983.

[Zave] P. Zave, "An Insider's Evaluation of PAISLey," *IEEE Transactions on Software Engineering*, Vol. 17, No.3, March, 1991 (pp.212-225).

GLOSSARY.

bags - a multiset, or a special ste which allows for duplicates of elements.

BagL - a language wherein the only object of computation and/or manipulation is an ordered bag of ordered bags.

primitive - a design element which must be present in a design in order to avoid ambiguity in a design.

Problem Solving Abstraction (PSA) - an environment which provides tools for the purpose of problem solving. Currently, the typical problem solving abstraction consists of the procedural programming language environment.

program generation - a mechanical production of executable programs from requirement or design specifications.

semantics - the meaning of a syntactic unit either formally or operationally.

semantics, logical - a logical interpretation of a transformed program statement to facilitate reasoning about the program.

semantics, program - an executable interpretation of a program statement.

software requirement - *constraints* of a problem to be solved.

software specification - a statement of the functionality intended of a software component.

software design - *how* a problem is to be solved.

syntax - the rules which state how to construct a valid sentence in some language.

synthesis - a formal method of program generation from specifications.

INDEX.

Appendix A - Statement of Work

We propose to investigate the theoretical and applied questions associated with establishing an unambiguous software design methodology. This work will contribute both to the areas of software engineering and automatic programming/design. In software engineering, the work will lead to a formal method for the development of a design methodology. In the area of automatic programming/design the proposed work will lead to a more precise method for system specification.

We intend to develop an unambiguous design methodology through the formal analysis of existing software design elements. As a result of the analysis we will identify a set of software design primitives. A design element will be a primitive element if and only if its absence from the design methodology leads to ambiguity. At the same time the remaining design elements will be deemed nonprimitive. Methods to derive the nonprimitive elements from the primitives will be developed.

Once the design methodology has been developed, work will begin on an automatic design tool. The automatic design tool will be based on the design methodology and will be written in Prolog. The specifications for the design tool will arise out of the methods of derivation defined for the nonprimitive elements.

Finally, to test the design methodology and automatic design tool, sample problem solutions will be submitted to the automatic design tool.

Appendix B -

Daniel E. Cooke, Ph.D.

GENERAL INFORMATION:

DATE OF BIRTH: March 23, 1955
CITIZENSHIP: U.S.

Education

1984-1986	Ph.D. in Computer Science, University of Texas at Arlington.
1977-1978	Master of Computing Science, Texas A&M University.
1973-1977	Bachelor of Science, Sam Houston State University.

Honors

MacIntosh-Murchison Chair in Engineering, MacIntosh-Murchison Faculty Fellow, Visiting Research Professor at Naval Postgraduate School (Summers, 1991, 1992), American Electronics Association Fellow, Sigma Xi, Tau Beta Pi, U.P.E., Alpha Chi.

Professional

9/92- present	MacIntosh-Murchison Chair in Engineering, Associate Professor of Computer Science, and Graduate Advisor, The University of Texas at El Paso.
9/91-8/92	MacIntosh-Murchison Faculty Fellow, Assistant Professor of Computer Science, and Graduate Advisor, The University of Texas at El Paso.
5/87-8/91	Assistant Professor of Computer Science and Graduate Advisor, The University of Texas at El Paso.
9/86-5/87	Assistant Professor of Computer Science, Texas Christian University.
5/86-9/86	Senior Software Engineer, Advanced Technology Department, General Dynamics, Corp.
9/84-5/86	Assistant Instructor of Computer Science, The University of Texas at Arlington.
5/82-5/84	Instructor of Computer Science, Hardin-Simmons University.
10/80-5/82	Systems Analyst, Data Processing Center, Texas A&M University.
1/79-10/80	Research Associate, Computer Science Department, Texas A&M University.
9/77-12/78	Graduate Assistant in Teaching, Computer Science Department, Texas A&M University.

RESEARCH:

Visiting Research Professor at Naval Postgraduate School (Summers 1991, 1992)

Research Contracts

Principal Investigator for a Contract: "The Formal Definition of an Inherently Unambiguous Design Methodology," U.S.A.F. Office of Scientific Research #F49620-89-C-0074. From June, 1989 - December 1992.

Principal Investigator for a Contract: "A Plan for a National Center for Computer Aided Software Engineering," U.S. Navy, Office of Naval Research # N60921-89-C-A182 From August, 1989 - March, 1990.

Co-PI Materials Research Center of Excellence, National Science Foundation.

Co-PI National Science Foundation CISE Computer Science Dept. UTEP.

Invited Talks

"Proving Properties of a CASE Tool," Texas A&M University, Hosted by Dr. John Leggett, April 16, 1990.

"Issues in Computer Aided Software Engineering," University of California, Berkeley, Hosted by Professor C.V. Ramamoorthy, April 30, 1990.

"Issues in CASE Technology Transfer," IEEE CASE '90 Fourth International Workshop, Irvine, California, December 6, 1990.

"The Development of a Requirement Specification Language," Texas A&M University, Hosted by Dr. John Leggett, February 6, 1991.

"Logic and Software Engineering," Panelist, Third International Conference on Software Engineering and Knowledge Engineering, June 1991.

"Ambiguity in Software Engineering," Panelist, Third International Conference on Software Engineering and Knowledge Engineering, June 1991.

"Expert Systems in Program Verification," Space Grant Consortium, Austin, Texas, June 18, 1991

"An Overview of CASE," Naval Postgraduate School, Hosted by Dr. Luqi, July-August, 1991.

"Program Synthesis: Generalizations for the Purpose of Program Synthesis," Naval Postgraduate School, Hosted by Dr. Luqi, July-August, 1991.

"Program Synthesis: Languages L2 and DecSpec Purpose of Program Synthesis," Naval Postgraduate School, Hosted by Dr. Luqi, July-August, 1991.

"Program Synthesis: Ambiguity Issues," Naval Postgraduate School, Hosted by Dr. Luqi, July-August, 1991.

"Is Artificial Intelligence the Answer for Software Engineering?" Panelist, Third IEEE Tools for Artificial Intelligence Conference, San Jose, CA, November, 1991.

"Software Automation Issues," 1992 IEEE International Conference on Systems Integration, Morristown, New Jersey, June 15, 1992.

"Novel Approaches to Systems Integration," 1992 IEEE International Conference on Systems Integration, Morristown, New Jersey, June 16, 1992.

"An Introduction to BagL," Naval Postgraduate School, Hosted by Dr. Luqi, August, 1992.

"An Application of CAPS to Support Software Maintenance," Naval Postgraduate School, Hosted by Dr. Luqi, August, 1992.

"Formal Methods in CASE," *IEEE CASE Fifth International Workshop*, Montreal, Canada, June 8, 1992.

"Software Engineering Support for Program Generation and Maintenance," Research Analysis and Maintenance, Hosted by Ray Day, August, 1992.

"Knowledge Acquisition and Process Acquisition," *IEEE Tools With Artificial Intelligence '92*, Arlington, Virginia, November, 1992.

Refereed Journal Publications

D.E. Cooke, "Formal Specifications of Resource-Deadlock Prone Petri Nets," *The Journal of Systems and Software*, Vol. 11 No. 1 (January, 1990) pp. 53-69.

D.E. Cooke, "Towards a Formalism To Produce a Programmer Assistant CASE Tool," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 2 No. 3, September, 1990, pp. 320-326.

D.E. Cooke and A. Gates, "On the Development of a Method to Synthesize Programs from Requirement Specifications," *International Journal on Software Engineering and Knowledge Engineering*, Vol 1 No 1, (March, 1991) pp. 21-38.

Daniel E. Cooke, "The Impact of CASE on Software Development Processes: Guest Editor's Introduction," *International Journal on Software Engineering and Knowledge Engineering*, Vol 1 No 2, (June, 1991) pp. iii-iv.

Daniel E. Cooke, "An Issue of the Next Generation of Problem Solving Environments," *Journal of Systems Integration*, Vol 1(2), (February, 1992) pp. 39-52.

Daniel E. Cooke, "The Impact of CASE on Software Development Processes II: Guest Editor's Introduction," *International Journal on Software Engineering and Knowledge Engineering*, Vol 2 No 2, (June, 1992) pp. 169-170.

C.V. Ramamoorthy, Daniel E. Cooke, and Chitta Baral, "Maintaining the Truth of Specifications in Evolutionary Software," to appear *International Journal of TAI*.

Journal Papers in Preparation.

Daniel E. Cooke, "Possible Effects of the Next Generation Programming Language on the Software Process Model," under revision.

Luqi and Daniel E. Cooke, "Rapid Prototyping and a Model for Software Maintenance," in preparation.
Daniel E. Cooke, "An Introduction to a High Level Programming Language," in preparation.

Refereed Conferences/Proceedings

D.E. Cooke, "Petri Nets: A Tool for Representing Concurrent Activities in Space Station Applications," *Space Station Automation III*, Wun Chiou, Sr., Editor, Proc. SPIE 851, pp. 53-63 (1987).

D.E. Cooke, "Proving Properties of Software Design Methods," *Proceedings of the First International Conference on Software Engineering and Knowledge Engineering*, June, 1989, pp.9-12.

A. Gates and D. Cooke, "An Introduction to the Recognition of Iterative Structures by a CASE Tool," *Proceedings of the Second International Conference on Software Engineering and Knowledge Engineering*, Skokie, Illinois, June, 1990 pp.201-208.

Daniel E. Cooke and Ann Gates, "On the Application of Stratification to Requirement Specifications," *Proceedings of the Second International IEEE Conference on Tools for Artificial Intelligence*, November, 1990, pp. 760-766.

D.E. Cooke and D. Patterson, "Towards a General Formula for Analogical Learning Leading to More Autonomous Systems," *Proceedings of SPIE: Intelligent Robots and Computer Vision IX: Algorithms and Techniques*, Vol. 1381, Ed. David Casasent, pp. 299-305 (1990).

D.E. Cooke, "Issues in CASE Technology Transfer," *Proceedings of IEEE Fourth International Workshop on Computer Aided Software Engineering*, Irvine, California, December 1990, pp 78-79.

Daniel E. Cooke, "Methods of Program Generation for Engineering Applications," *Proceedings of ASME Energy-sources Technology Conference* , Houston, Texas (January, 1991), pp. 15-20.

Daniel E. Cooke, Miguel Pedroza, and Ann Gates, "Interaction Of Data Structures and Primitive Operations of Language LO," *Proceedings of the Third International Conference on Software Engineering and Knowledge Engineering*, June 1991, pp. 78-83.

C.V. Ramamoorthy and Daniel E. Cooke, "The Correspondence Between Methods of Artificial Intelligence and the Production and Maintenance of Evolutionary Software," *Proceedings of the Third International IEEE Conference on Tools for Artificial Intelligence*, November, 1991, pp. 114-118.

Ann Gates and Daniel E. Cooke, "On a Fundamental Relationship Between Software Reuse and Software Synthesis," *Proceedings of Hawaii International Conference on System Sciences Vol. II*, Kauia, Hawaii (January, 1992) pp. 539-548.

Mike Pedroza and Daniel Cooke, "The Informal Semantics of BagL," *PD-Vol. 43, Computer Applications and Design Abstraction ASME 1992*, Houston, Texas (January, 1992) pp. 29-32.

John F. Kennedy and Daniel E. Cooke, "An Application of 3GL Design Principles to Explain 4GL Maintenance Difficulties," *PD-Vol. 43, Computer Applications and Design Abstraction ASME 1992*, Houston, Texas (January, 1992) pp. 129-133.

Luqi and D. Cooke, "The Management of Uncertainty in Software Development," *IEEE COMPSAC '92*, Chicago, IL, pp. 381-386.

Daniel E. Cooke, "Issues Surrounding Specification Languages For Software Automation," *Proceedings of IEEE Fifth International Workshop on Computer Aided Software Engineering*, July 6-10, 1992, Montreal, Canada, pp. 120-123.

Daniel E. Cooke and Aida Gutierrez, "An Introduction to BagL," *IEEE Fourth International Conference on Software Engineering and Knowledge Engineering*, Capri, Italy, pp. 479-486.

Daniel E. Cooke, "Logical Development of a Petri Net Deadlock Analysis Program," *Proceedings of the Fourth International IEEE Conference on Tools for Artificial Intelligence*, November, 1992, pp. 230-233.

Daniel E. Cooke, "Arithmetic Over Multisets Leading to a High Level Language," *Proceedings Computers in Engineering Symposium 1993*, Houston, Texas (January, 1993) to appear.

Panel Papers/ Reviews

D. Cooke, T. Escamilla, and M. Gibson, "The Correspondence Between Methods of Artificial Intelligence and the Production and Maintenance of Evolutionary Software," *Proceedings of the Third International Conference on Software Engineering and Knowledge Engineering*, June 1991, pp. 114-115.

B. Blum, D. Cooke, X. Li, N. Minsky, and R. Semmel, "The Best Approach to Knowledge Representation for Software Engineering," *Proceedings of the Third International Conference on Software Engineering and Knowledge Engineering*, June 1991, pp. 166-167.

D. Cooke, M. Feather, S. Fickas, N. Minsky, P. Selfridge, D. Smith, and J.P. Tsai, "Is AI the Solution for Software Engineering?," *Proceedings of the Third International IEEE Conference on Tools for Artificial Intelligence*, November, 1991, pp. 10-12.

Daniel E. Cooke, "Review of Software Conflict: Essays on the Art and Science of Software Engineering," *International Journal on Software Engineering and Knowledge Engineering*, Vol 1 No 4, (December, 1991) pp. 477- 478.

Valdis Berzins, Daniel E. Cooke, Luqi , Peter Ng, C.V. Ramamoorthy, Murat Tanik, Joe Urban, and Raymond Yeh, "Workshop on Software Automation," *IEEE Systems Integration Conference, Proceedings of 1992 IEEE International Conference on Systems Integration*, Morristown, NJ, (June 15-19, 1992) pp. 720-722.

Educational Publications

D.E. Cooke, S.A. Starks, and D.S. Thorp, "CSAD: A Course Advisor," ASEE CoED Journal, Vol. VIII No. 4 (October-December, 1988, pp. 71-75.

D.E. Cooke, S.A. Starks, and A.F. Rodriguez, "A Methodology for Computer Assisted Learning Using Expert Systems," ASEE CoED Journal, Vol. VIII No. 4 (October-December, 1988, pp. 38-42.

D.E. Cooke, S.A. Starks, and D.S. Thorp, "CSAD: A Course Advisor," Proceedings: Engineering Focuses on Excellence. American Society of Engineering Education, 1987. pp. 658-663.

D.E. Cooke, S.A. Starks, and A.F. Rodriguez, "A Methodology for Computer Assisted Learning Using Expert Systems," Proceedings: Engineering Focuses on Excellence. American Society of Engineering Education, 1987. pp. 1481-1485.

Technical Reports

"Towards a Formalism for Program Generation," Daniel E. Cooke, for the Air Force Office of Scientific Research, *F49620-89-C-0074, July, 1992.

"Towards a Formalism for Program Generation," Daniel E. Cooke, for the Air Force Office of Scientific Research, *F49620-89-C-0074, July, 1991.

"Towards a Formalism for Program Generation," Daniel E. Cooke, for the Air Force Office of Scientific Research, *F49620-89-C-0074, July, 1990.

"A Plan for a National Center of Excellence for Computer Aided Software Engineering," D.E. Cooke, for the Naval Surface Warfare Center * N60921-89-C-A182, March, 1990.

"Effective Analogical Learning," D.W. Patterson and D.E. Cooke. NASA - JSC - Grant * NAG 9-285: 1988.

"Computer Network Design," D.E. Elizandro, D.E. Cooke, et al. State of Texas: 1986.

"Case Tracking Users Manual," D. E. Elizandro and D.E. Cooke, State of Texas: 1986.

"RADC Strategic Defense Initiative Battle Management C3 Technology Program: Technical Description Document," States Nelson, D.E. Cooke, and S. Madaras. Rome Air Development Center for Candidate High_Payoff Tools. September, 1986.

Texts

Logic: The Basis for Understanding Prolog. ABLEX Advanced Topics in Computer Science, edited by Udo Pooch. In Press.

The Impact of Computer Aided Software Engineering on Software Processes. World Scientific Publishers, Ltd. Contributors: Raymond Yeh, Peter Ng, Luqi, Joseph Urban, Ron Norman, W.D. Hurley, John Baker, Patrick Bobbie, WT Tsai, Greg Boone, Nick Bourbakis, etc. In Press.

Software Automation. Contributors: CV Ramamoorthy, Raymond Yeh, Peter Ng, Luqi, Berzins, Joseph Urban, Murat Tanik, etc. In progress.

SERVICE

To the Profession:

Editorships and Program Committees:

Second International Conference on Software Engineering and Knowledge Engineering (S.K. Chang, Conference Chair).
Third International Conference on Software Engineering and Knowledge Engineering (S.K. Chang, Conference Chair).
Fourth International Conference on Software Engineering and Knowledge Engineering (S.K. Chang, Conference Chair).
Second IEEE Systems Integration Conference (Raymond Yeh and Peter Ng, Conference Co-Chair).
Third IEEE Systems Integration Conference (Raymond Yeh and Peter Ng, Conference Co-Chair).
European Joint Conference on Engineering Systems Design and Analysis (A. Ertas and T. Derbentli, Conference Co-chair).
IEEE TAI '91 (Benjamin Wah Conference Chair).
IEEE TAI '92 (Stephanou Conference Chair).

Associate Editor of the *Journal for Software Engineering and Knowledge Engineering.*
Book Review Editor of the *Journal for Software Engineering and Knowledge Engineering.*
Co-editor *COMPUTER APPLICATIONS AND DESIGN ABSTRACTION 1992 - ASME.*

Vice Program Chair *IEEE TAI '92.*
Symposium Chair *ASME Computer Applications and Design Abstraction '93.*
Chair of Workshop on Software Automation for Systems Integration Conference 1992.
Chair of Workshop on Software Automation for Systems Integration Conference 1993.

Reviewing Activity

Journals:

IEEE Computer, IEEE Transactions on Knowledge Engineering and Data Engineering, IEEE Software, International Journal of Software Engineering and Knowledge Engineering, International Journal on Systems Integration, and Journal of Systems and Software.

Conferences:

Third IEEE Systems Integration Conference
1989 Hawaii International Conference on System Sciences.
1992 Hawaii International Conference on System Sciences.
Second International Conference on Software Engineering and Knowledge Engineering.
Third International Conference on Software Engineering and Knowledge Engineering.
Fourth International Conference on Software Engineering and Knowledge Engineering.
European Joint Conference on Engineering Systems Design and Analysis.
Organized Panels for ICSEKE and IEEE Tools for AI Conference.
Local Arrangements for Regional ACM Conference in Feb. 1990.
IEEE TAI '91.
IEEE TAI '92.

To The University:

Intellectual Property Committee (1989-present)
Ph.D. proposal committee for Computer Engineering (1987-1988)
University Computer Center Director Search Committee (1991)
Dean of the College of Business Search Committee (1991)
Liberal Arts Computer Center Director Search Committee (1991)
ISIS Replacement Committee (1992-present)
Chair of Faculty Task Force for ISIS Replacement (1992-present)
Asst. Vice President for Intructional Technology Search Committee (1992)
Faculty Advisor to UPE and ACM (1987-1990)

Graduate Advisor (1988-present)

To The Community:

Elder of University Presbyterian Church.
Board of Directors INSIGHTS Science Museum.
Steering Committee of Paso Del Norte National Issues Forum.
Moderator for Paso Del Norte National Issues Forum.