A CHARACTERIZATION OF FUNCTIONS FOR

EXECUTION TIME COST ANALYSIS

IN FP

By

RICHARD WALTER MATZEN

Bachelor of Science
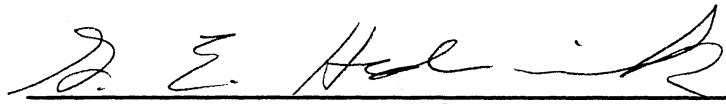
University of Central Arkansas

Conway, Arkansas

1984

Submitted to the faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
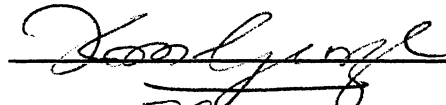for the Degree of
MASTER OF SCIENCE
July, 1987

A CHARACTERIZATION OF FUNCTIONS FOR

EXECUTION TIME COST ANALYSIS

IN FP

Thesis Approved:

_____
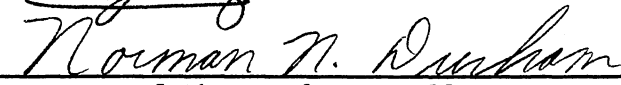Thesis Adviser

_____

_____
Dean of the Graduate College

PREFACE

A method for characterizing functions in FP, a functional programming language, was developed to support execution time cost analysis. A set of restrictions of each function in FP is defined which corresponds to the possible computation sequences of the function. Then a method is shown to construct equations for the domain and the range of each restriction. Proofs are given that the method is correct and examples are shown. A program was written to implement the method and results of program execution are shown in table form. The results show that the method can be used to estimate the execution time cost of f over the data domain, O. A subset of FP is considered which includes functions for condition, construction, and composition.

I wish to express my gratitude to all of the people who have assisted me in this work. In particular, I would like to thank my major advisor, Dr. G. E. Hedrick, for his guidance and support in completing my course of study, and for his observations and advice which were helpful in developing this paper.

I am especially indebted to Dr. K. M. George. Through his expertise in the field of study, FP, and generous contributions of his time he has contributed significantly to the rigor of this paper. I would also like to thank Dr. D. D. Fisher for his helpful advice on machine models and cost analysis.

iii

I could not have completed this course of study without the help of my family. I would like to thank my father and my mother for their assistance and for their encouragement. Last, but certainly not least, I am grateful to my wife, Jan, and my two sons, Mika and Ben. Their moral support, their patience, and their faith in me have kept me going.

TABLE OF CONTENTS

LIST OF SYMBOLS

!	– Logical or
&	– Logical and
==>	– Logical implication
U	– Union of sets
∩	– Intersection of sets
=	– Strong equality
φ	– The empty set
∈	– Set containment
∉	– Not in the set
F	– Set of functions in FP (Def. 3.2, page 9)
f, f1, fn	– Functions in F
G	– Set of functional forms in F (page 4)
O	– Set of objects in FP (page 4)
⊥	– Bottom or "undefined" in FP (page 5)
●	– Functional form for composition (App. A, page 66)
[]	– Functional form for construction (App. A, page 66)
-->	– Functional form for condition (App. A, page 66)
x	– Constant function, x, where x is an object in O (App. A, page 66)
f_i	– The i'th first order restriction of f where f is any function in F (Def. 3.4, page 11)
flast	– The number of first order restrictions of f (Def. 3.2, page 9)

$d(f_i)$	— The domain of f_i (Def. 3.4, page 11)
$r(f_i)$	— The range of f_i (Def. 3.4, page 11)
f_i^c	— The i'th computational restriction of f where f is any function in F (Def. 3.5, page 12)
$d(f_i^c)$	— The domain of f_i^c (Def. 3.5, page 13)
$r(f_i^c)$	— The range of f_i^c (Def. 3.5, page 13)
$p(f)$	— The number of computational restrictions of f (Def. 3.5, page 12)
D^c	— The domain of the inverse set mappings for the computational restrictions of functions in FP (Def. 3.7, page 22)
D_1^c	— The domain of the inverse set mappings for the computational restrictions of functions in FP (Def. 3.7. page 22)
UD^c	— The finite unions of sets in D^c (Def. 3.7, page 23)
UD_1^c	— The finite unions of sets in D_1^c (Def. 3.7, page 23)
$<D_1,..D_n>$	— The closed form of sets in D^c (Def. 3.7, page 23)
X	— The name of the object x in O (Def. 3.9, page 36)
$X.i$	— The name of the i'th child of X (Def 3.9, page 36)
$cost(f:x)$	— The execution cost for a function f in F with an object x (Def. 4.1, page 52)
$c(f)$	— A symbolic constant for the execution cost of any function f in F (Def. 4.1, page 52)

CHAPTER I

INTRODUCTION

Statement of Problem

Formal techniques for estimating program execution time by analysis
of the program text have been demonstrated for conventional procedural
languages. Results are generally expressed as the cost of executing the
program over all possible inputs for the program. Estimates of
execution time are useful for comparing the efficiency of various
versions of a program and for input to resource allocation schemes.
Partially automated techniques for this type of cost analysis have been
demonstrated successfully for programs containing iteration but not for
the general case.

FP, a functional programming language, differs from conventional
languages in several ways: there are no variables or assignment
statements, the data environment consists of a single structured object,
and all functions are built recursively from simpler functions. Also,
each function in FP is defined by a conditional expression and thus, a
conditional branch is possible at each step in the computation.

Due to these differences it is not apparent that existing
techniques are either applicable or sufficient for estimating execution
time of FP functions. The questions considered in this thesis are :
Does there exist a process similar to those described in the literature
for conventional languages which can be used to estimate the execution

1

time of FP functions? If so, can the process be automated and for what level of programming complexity will it work? Also, does a method exist to determine the possible computation sequences of functions in FP? If so, can this method be used to provide a framework for estimating the execution time cost of functions over all possible inputs?

Objectives

In this thesis a method is developed and demonstrated for estimating the execution time cost of FP functions. The method is similar to existing methods of cost analysis.

1) Execution time cost is expressed in terms of the number of basic operations performed.

2) A computational model is defined for FP.

The computational model defines the execution time cost of each function for each possible input. However, due to the absence of program variables in FP existing methods give no apparent solution to the problem of estimating the execution time cost of a function over all possible inputs. To solve this problem a characterization of each function is given. This characterization of functions provides a framework for execution time cost analysis in FP.

Chapter 2 contains background and definitions for FP and execution time cost analysis and a review of the literature for this thesis. In Chapter 3 a characterization of functions is shown for FP. A set of restrictions is defined for each function and equations are derived from the induced set mappings of the restrictions are constructed for the domain and range of each restriction. Proofs are given that the equations are correct and examples are shown. In Chapter 4 a proof is

given that the set of restrictions of the function corresponds to the computation sequences of the function. Also, a computational model is defined for FP and methods are shown to compute the execution time cost associated with each restriction. The symbols and notation used in this paper are defined in the List of Symbols on page vii.

A program was written in the 'C' programming language to implement the above described methods, and a table of program results is given for various cases of functions in FP. The domain, range, and cost estimate for each restriction of the functions are listed in Appendix H. Due to the size of the program the source code is not included in this thesis. Copies of the program may be obtained by writing to the following address.

Course Record Files: Richard W. Matzen

Department of Computing and Information Sciences

M.S. 219

Oklahoma State University

Stillwater, Oklahoma, 74078

Some of the results listed were computed by hand. These cases are duly noted.

Since this is an initial study of execution time cost analysis for FP, only a subset of FP is considered. The subset is defined in Chapter 3 and Appendix A.

CHAPTER II

LITERATURE REVIEW

FP

Backus (3) presents a case for the development of functional programming languages as an alternative to conventional algebraic languages. A broad class of languages is outlined which are based on functional forms and primitive functions. These functions are used to build function expressions which are used as arguments to other functional forms, the result being a strictly hierarchical programming system with no side effects. A model is presented for a class of a functional programming systems which are called FP systems. This model is later referred to as the language, FP (4).

The semantics of FP is defined as follows (3). FP consists of the following:

1) A set O of objects. An object is either an atom, a sequence of finite length whose elements are objects, or $\perp$ ('undefined' or 'bottom'). An atom may be any string of capital letters, digits, and special symbols and therefore may be a number, a string, or T or F representing boolean values.

2) A set P of primitive functions.

3) A set F of functions that map objects into objects.

4) A set G of functional forms which form new functions by taking existing functions as arguments.

5) A set D of definitions that define functions in F and assign each a name.

A function f in F may be :

a) A primitive function.

b) A functional form with functions in F as arguments.

c) A definition of the form f = d, where d is a function in F.

d) None of the above, in which case f:x is $\bot$.

The specific primitives and functional forms will not be listed but can be easily obtained from the literature (3), (4), (11). The subset of F considered in this paper is defined in Appendix A.

A program in FP has a single operation, application, in which a function is applied to an object. All functions f in F are strict. That is, f:$\bot$=$\bot$, for all f. Also, any sequence containing $\bot$ is defined to be $\bot$. There are two cases where f:x may be $\bot$. One case occurs when the computation of f:x terminates and yields $\bot$ and the second case occurs when f:x is nonterminating.

Execution Time Cost Analysis

There are several approaches to the problem of estimating execution time cost (12). Cost may be estimated by measurement, either by actual time or by counting the number of operations performed during execution of the program with various inputs. Either process is an estimation. Results for actual time will vary according to system performance and counting mechanisms generally only count certain relevant operations.

Another approach is to estimate the execution time cost by analysis of the program text. In this case cost must be expressed in terms of the number of basic operations performed and variables which represent

the size of inputs which affect the number of times these operations are performed. Since specific input values can affect the time required for each primitive operation the typical method of estimating is to assign a constant cost to each primitive operation (12). The results lose some of their accuracy but gain independence from any particular computing system.

The general approach in existing methods for cost analysis is to develop a computational model for executing programs in the specified language (1), (12). The model determines a cost formula or rule for the cost of executing each construct in the language and a method for determining the computation sequence of a given program. The cost of a program is the sum of the costs of executing each statement in the computation sequence. Thus, the model gives a cost estimate for each possible set of inputs to the program.

A difficult task in execution time cost analysis is estimating the cost of program execution over the entire data domain (1). One approach to this problem involves giving cost expressions containing program variables which represent the size of relevant inputs (12). However, this frequently fails to provide a single cost expression for the cost of the function over the data domain. Then some basis must be established for the cost estimate over the data domain. Typical approaches are to give an expression for best, worst, or average case cost (1).

Another problem encountered is formally proving the correctness of execution time cost estimates. One method uses axiomatic semantics to prove assertions about the number of times certain operations are performed. It is limited in scope and potential for automation because it depends on an externally developed set of assertions (12).

Type Inference Schemes

One comprehensive approach to reasoning about programs is called type inference, where type information about some program structure is determined by analysis of the program text (2). Type is defined for functions in terms of the types of the domain and the range of the function.

A technique for type inference called reduced computation has been developed and applied to a subset of FP by Katayama (7). First a set of types is defined. Then a method is given for determining the type of any function in terms of the types of its domain and range. The type computations are determined directly from the definitions of the primitive functions and functional forms. The method is based on relations which map the type of arguments to the type of results. For each function f in F a relational expression, f', can be derived which performs the reduced computation and gives the type of f. Type information is represented by type expressions for the structure and primitive types of components. This is a typical approach to defining types for type inference schemes (1).

CHAPTER III

A CHARACTERIZATION OF FUNCTIONS IN FP

Definitions and Preliminaries

Various methods for characterizing programs by analysis of the program text are described in Chapter 2. Some of these methods are useful for estimating the execution time cost of programs over all possible inputs (1), (12). However, none of the methods described are applied to the problem of execution time cost analysis for functional languages.

In FP all program operations are the application of some function f in F to some object x in O. Thus, it is reasonable to assume that a characterization of f which describes the domain and range of f might be useful for execution time cost analysis. The characterization must be nontrivial since for all f in F the domain of f is O. In this chapter a set of restrictions of f is defined which has the following properties.

1) The set of restrictions is a finite set.

2) The domains of the restrictions of the set partition, O, the domain of f.

3) The domain and range of each restriction can be derived.

4) The computation sequence of f:x is the same for all x in the domain of each restriction in the set.

The set, F, considered in this chapter is a subset of the set of functions defined for FP in Chapter 2.

8

In this section the set of restrictions is defined and properties 1 and
2 are proved. In Section 2 property 3 is proved and equations are given
for the domains and ranges of the restrictions. In the following
chapter property 4 is proved and the characterization of f given by
properties 1-4 is shown to be useful for estimating the execution time
cost of functions f in F over the data domain, 0.

Definition 3.1. If f and g are functions, f:X-->Y and g:X-->Y, then f
and g are equivalent functions iff

    for all x in X, f:x = g:x.

Definition 3.2. For the purposes of this paper the set F is defined as
follows.

    A) A function f in F is defined to be (3):

    for all x in 0,

        $f:x = P_1(x) \longrightarrow f_1; \ldots; P_{last-1}(x) \longrightarrow f_{last-1}; f_{last}.$

    where f:x is evaluated by (9):

        if $P_1(x)$ then $f:x = f_1:x$

        if $\sim P_1(x)$ & $P_2(x)$ then $f:x = f_2:x$

        . . . .

        if $\sim P_1(x)$ & ... & $\sim P_{last-1}(x)$ then $f:x = f_{last}:(x)$

    To simplify notation let 'P_{last}' denote '$\sim P_1$ & ... & $\sim P_{last-1}$'.

    B) For all f in F defined in A) above, if

        $\{x| P_i(x) = True\} \cap \{x| P_j(x) = True\} \neq \phi$, for some $i \neq j$

    then replace f by an equivalent function f' which is derived from f

    by replacing P_j with P'_j where $P'_j = P_j$ & $\sim P_i$.

    C) The set F is limited to the functions defined in Appendix A.

The definitions in Appendix A are from Backus (3) and Williams (11) except for [], atom, and null. For these cases the definitions given are for equivalent functions which are derived for the purposes of this paper. The equivalent functions for null and atom are defined in Definition 3.2.B. The motivation for this extended definition is to ensure that for all f in F the following property holds:

$$\{x \mid P_i(x)\text{=TRUE}\} \cap \{x \mid P_j(x)\text{=TRUE}\} = \emptyset$$

for $i,j\text{=}1,\ldots,$flast, $i \neq j$.

The methods of this chapter are based on this property of F. The motivation for defining the equivalent function for [] is shown later. For all cases proof of equivalence is direct by Definition 3.1 and the definitions in Appendix A.

The following notation is used in the remainder of this paper.

1) BOOL denotes $\{x \mid x\text{=}T \mathbin{!} x\text{=}F\}$

2) NUM denotes $\{x \mid x$ is a number$\}$

3) ATOM denotes $\{x \mid x$ is an atom$\}$

4) R denotes $\{eq,\leq,<,\geq,>\}$

A complete list of symbols and notation used in this paper is given in the List of Symbols on page vii. In Appendix B computations of f:x are shown for simple cases of f and x. An understanding of these computations is necessary to understand the methods and proofs of this chapter. For the sake of simplicity the definitions of f $\in$ R in Appendix A are different from those given in the original papers (3), (11).

Definition 3.3. For any function f, f:X --> Y, a function g is a restriction of f if the domain of g is contained in X, and for all x in the domain of g, g:x = f:x.

Definition 3.4.

A) For a function f in F defined as:

$$f:x \;\underline{=}\; P_1 \longrightarrow f_1; \ldots; f_{last},$$

let the first order restrictions of f be defined as:

$$P_i \longrightarrow f_i \;, \quad i=1,\ldots,flast.$$

The first order restrictions will be denoted simply by 'f_i' and are derived directly from the syntax of f.

B) For all f and for all f_i, $i=1,\ldots,flast$ let $d(f_i)$ denote the domain of f_i and $r(f_i)$ denote the range of f_i. Then

$$d(f_i) = \{x \mid P_i(x)=\text{True }\} \text{ and}$$

$$r(f_i) = f_i:d(f_i).$$

Example 3.1.

1) Consider the function f defined by $f:x\underline{=}tl:x$. The first order restrictions of f are:

$$tl_1:x \;\underline{=}\; (x=<x_1>) \longrightarrow <>$$

$$tl_2:x \;\underline{=}\; (x=<x_1,\ldots,x_n>,\; n\geq 2) \longrightarrow <x_2,\ldots,x_n>$$

$$tl_3:x \;\underline{=}\; {\sim}(x=<x_1,\ldots,x_n>,n\geq 1) \longrightarrow \underline{\mathsf{!}}$$

2) Consider any function $f:x=[f1,\ldots,fn]:x$ where $f1,\ldots,fn$ are in F. The first order restrictions of f are:

$$[f1,\ldots,fn]_1:x \;\underline{=}\; {\sim}(f1:x = \underline{\mathsf{!}} \;!\ldots!\; fn:x = \underline{\mathsf{!}}) \longrightarrow <f1:x,\ldots,fn:x>$$

$$[f1,\ldots,fn]_2:x \;\underline{=}\; (f1:x=\underline{\mathsf{!}} \;!\ldots!\; fn:x=\underline{\mathsf{!}}) \longrightarrow \underline{\mathsf{!}}$$

Proposition 3.1. For all f in F the following properties hold:

1) $d(f_i)$, $i=1,\ldots,flast$ partition O.

2) $r(f_i) \cap r(f_j) = \emptyset$, for all $i \neq j$, $i,j=1,\ldots,flast$.

Proof: The proof that these properties hold is direct by Definitions 3.2.B and 3.4.B.

The first order restrictions only give a trivial description of f in the general case. However, a set of restrictions of f in F which satisfy properties 1-4 outlined in the beginning of this chapter can be derived from the first order restrictions. The following definition gives the derivation of this expanded set of restrictions of f.

<u>Definition 3.5.</u> The computational forms of f are denoted by f_j^C, j=1,...,p(f), where p depends on f and are defined below for the various cases of f in F.

A) For all f in F and for all x in O the computational forms of f are defined as follows:

1) If f is a primitive function in f or f is the constant functional form, then

$$f_i^C:x = f_i:x, \quad i=1,...,f\text{last}$$

where f_i are the first order restrictions of f.

2) If f = G(f1,...,fn) where G is a functional form, then the computational forms of f are defined in terms of the first order restrictions of f and the computational forms of f1,...,fn.

a) if f=(f1⊛f2) then the computational forms of f are:

$$f_i^C:x = (f1_j^C \circledast f2_k^C)_1:x$$

for i=1,...,p(f1) X p(f2),   j=1,...,p(f1), k=1,...,p(f2).

b) if f=[f1,...,fn] then the computational forms of f are:

$$f_{i,1}^C:x = [f1_j^C,...,fn_k^C]_1:x$$

for i=1,..p(f1) X...X p(fn), j=1,..p(f1) ,..., k=1,..p(fn)

$$f_{i,2}^C:x = Q_1(f1_j^C:x=\underline{\perp}) \&...\& Q_n(fn_k^C:x=\underline{\perp}) \longrightarrow \underline{\perp}$$

for all $(Q_1,...,Q_n)$ where Q_i=True or Q_i=False, i=i..n, Q_i=True for at least one i, and for all i=1,...,((2p(f1) X...X 2p(fn)), j=1,...,p(f1) ,..., k=1,...,p(fn).

c) If $f=(f1-->f2;f3)$ then the computational forms of f are:

$$f_{i,1}{}^C:x \equiv (f1_j{}^C-->f2_k{}^C;f3)_1:x$$

　　　　for all i=1,...,(p(f1) X p(f2)), j=1,...,p(f1),

　　　　k=1,...p(f2).

$$f_{i,2}{}^C:x \equiv (f1_j{}^C-->f2;f3_k{}^C)_2:x$$

　　　　for all i=1,...,(p(f1) X p(f3)), j=1,...,p(f1),

　　　　k=1,...p(f3).

$$f_{i,3}{}^C:x \equiv (f1_j{}^C-->f2;f3)_3:x$$

　　　　for all i=1,...,p(f1).

3) Only what is defined above is a computational form of f.

B) For all f in F and for all $f_j{}^C$, j=1,...,p(f) defined in A above $d(f_j{}^C)$ is defined as follows for the various cases of f:

1) If f is a primitive function in F, then

$$d(f_j{}^C) = d(f_j) = \{x| P_j(x)=True\}.$$

2) If $f_j{}^C:x=(f1_i{}^C \circledast f2_k{}^C)_1:x$, then

$$d(f_j{}^C) = \{x| x \in d(f2_k{}^C) \ \& \ f2_k{}^C:x \in d(f1_i{}^C) \}$$

3) If $f_j{}^C:x = [f1_i{}^C,...,fn_k{}^C]_1:x$, then

$$d(f_j{}^C) = \{x| P_1(x)=True \ \& \ x \in (d(f1_i{}^C) \cap ... \cap d(fn_k{}^C)) \}$$

4) If $f_j{}^C:x = [f1_i{}^C,..fn_k{}^C]_2(Q_1,...,Q_n):x$, then

$$d(f_j{}^C) = \{x| Q_1(f1_i{}^C:x=\underline{i}) \ \&...\& \ Q_n(fn_k{}^C:x=\underline{i})$$
$$\& \ x \in (d(f1_i{}^C) \cap ... \cap d(fn_k{}^C))\}$$

5) If $f_j{}^C:x=(f1_i{}^C-->f2_k{}^C;f3)_1:x$, then

$$d(f_j{}^C) = \{x| P_1(x)=True \ \& \ x \in (d(f2_k{}^C) \cap d(f1_i{}^C)) \}$$

6) If $f_j{}^C:x=(f1_i{}^C-->f2;f3_k{}^C)_2:x$, then

$$d(f_j{}^C) = \{x| P_2(x) \ \& \ x \in (d(f3_k{}^C) \cap d(f1_i{}^C)) \}$$

7) If $f_j{}^C:x=(f1_i{}^C-->f2;f3)_3:x$, then

$$d(f_j{}^C) = \{x| P_3(x)=True \ \& \ x \in d(f1_i{}^C)\}$$

C) for all f in F and for all f_j^C, j=1,...,p(f)

$r(f_j^C) = f_j^C:d(f_j^C).$

The definition given above gives a set of forms for f in F which are shown to be restrictions of f in the following theorem.

<u>Theorem 3.1.</u> For all f in F the computational forms of f are restrictions of f.

<u>Proof:</u> Let f be in F and f_j^C, j=1,...,p(f) be the computational forms of f. To prove that f_j^C is a restriction of f it must only be shown for all x in O, that

$x \in d(f_j^C)$ ==> $f_j^C:x = f:x.$

For primitive functions f in F this only requires the observation that

$x \in d(f_j^C)$ ==> $P_j(x) =$ True.

For f=G(f1,...,fn) a proof by induction is given.

Let S(N) be the statement "If f=G(f1,...,fn) and f is defined by at most N applications of definitions of the functional forms in G, then

$x \in d(f_j^C)$ ==> $f_j^C:x = f:x.$"

Three cases of Definition 3.5.B.2 are proved. The remaining cases are similar.

Basis: If N=1, then f1,...,fn are primitive functions in F.

case 2) Let $f_j^C:x = (f1_i^C \oplus f2_k^C)_1:x$ and $x \in d(f_j^C)$.

==> $x \in d(f2_k^C)$ & $(f2_k^C:x) \in d(f1_i^C)$.   Df. 3.6.B.2

Since f1 and f2 are primitive functions,

==> $(f2_k^C:x=f2:x)$ & $(f1_i^C:(f2_k^C:x)=f1:(f2_k^C:x))$

Thus, $f_j^C:x = f:x$ if $x \in d(f_j^C)$.

case 4) Let $f_j^C:x = [f1_i^C,...,fn_k^C]_2:x$

$x \in d(f_j^C)$ ==> $Q_1(f1_i^C:x=\underline{\iota})$ &...& $Q_n(fn_k^C:x=\underline{\iota})$

& $x \in (d(f1_i^C) \cap...\cap d(fn_k^C))$ Df. 3.5.B.3

==> $P_2(x)$=True & x $\in$ (d(f1$_1^C$) $\cap$...$\cap$ d(fn$_K^C$))

==> $P_2(x)$=True & f1$_1^C$:x=f1:x &...& fn$_K^C$:x=fn:x

Thus, f$_J^C$:x = f:x if x $\in$ d(f$_J^C$).

case 5) Let f$_J^C$:x = (f1$_1^C$-->f2$_K^C$;f3)$_1$:x.

x $\in$ d(f$_J^C$) ==> $P_1(x)$=True & x $\in$ (d(f2$_K^C$) $\cap$ d(f1$_1^C$))   Df. 3.5.B

==> $P_1(x)$=True & f2$_K^C$:x=f:x

Thus, f$_J^C$:x = f:x if x $\in$ d(f$_J^C$).

Therefore, S(1) is true.

Inductive Step: Suppose S(N) is true for any N$\geq$1. Then if

f=G(f1,...,fn) is defined by N+1 applications of the definitions of the

functional forms in G, f1,...,fn are each defined by some M$\leq$N

applications of the definitions of the functional forms in G.

case 2) Let f$_J^C$:x = (f1$_1^C$$\circ$f2$_K^C$)$_1$:x and x $\in$ d(f$_J^C$).

==> x $\in$ d(f2$_K^C$) & ((f2$_K^C$:x) $\in$ d(f1$_1^C$))   Df. 3.5.B.2

Then, by the above observations about f1,...,fn,

f2$_K^C$:x=f2:x & f1$_1^C$:(f2$_K^C$:x)=f1:(f2$_K^C$:x).

Thus, f$_J^C$:x = f:x if x $\in$ d(f$_J^C$).

case 4) Let f$_J^C$:x=[f1$_1^C$,...,fn$_K^C$]$_1$:x & x $\in$ d(f$_J^C$).

==> Q_1(f1$_1^C$:x=$\underline{\perp}$) &...& Q_n(fn$_K^C$:x=$\underline{\perp}$)

& x $\in$ d(f1$_1^C$) &...& x $\in$ d(fn$_K$).

==> $P_2(x)$=True & x $\in$ (d(f1$_1^C$) $\cap$...$\cap$ d(fn$_K^C$))

Then by the above observations about f1,...,fn,

f1$_1^C$:x=f1:x &...& fn$_K^C$:x=fn:x.

Thus, f$_J^C$:x = f:x if x $\in$ d(f$_J^C$).

case 5) Let f$_J^C$:x = (f1$_1^C$-->f2$_K^C$;f3)$_1$:x.

x $\in$ d(f$_J^C$) ==> $P_1(x)$ & x $\in$ d(f2$_K^C$)   Df. 3.5.B

==> $P_1(x)$ & f2$_K^C$:x=f:x

Thus, $f_j{}^c:x = f:x$ if $x \in d(f_j{}^c)$.

Hence, by induction S(N) is true for all N≥1.

Therefore, the computational forms of f in F are restrictions of f.

Theorem 3.2 For all f in F, the computational restrictions of f is a finite set.

Proof: The number of computational restrictions of f, p(f), is equal to:

$$m_1 + \ldots + m_{flast} \quad \text{where each } m_i, \; i=1,\ldots,flast$$

is the number of computational restrictions induced by f_i. When f is a primitive function in F, $m_i = 1$, $i=1,\ldots,n$ and p(f)=flast. For f=G(f1,...,fn) a proof by induction is given.

Let S(N) be the statement, "If f is defined by at most N applications of the definitions of the functional forms in G, then $f_j{}^c$, j=1,...,p(f) is a finite set."

Basis: If N=1 then f1,...,fn are primitive functions in F. The computational restrictions induced by each f_i, are in the form $G_i(f1_j{}^c,\ldots,fn_k{}^c)$. Then the number of restrictions induced by $f_i = m_i$, which is equal to the number of permutations of (j,...,k). Since f1,...,fn are primitive functions, p(f1),...,p(fn) are finite and $f_j{}^c$, j=1,...,p(f) is a finite union of finite sets.

Thus, S(1) is TRUE.

Inductive Step: Suppose S(N) is TRUE for any N≥1. Then if f is derived by at most N+1 applications of the definitions of the functional forms in G, f1,...,fn must each be derived by some M≤N applications of the definitions of the functional forms in G and S(N) holds for f1,...,fn. Then by the same argument given for N=1, p(f) is finite.

Therefore, the computational restrictions of f are a finite set.

Definition 3.6 Any function g, g:X --> Y, induces the following two set

mappings (10):

for all A $\subseteq$ X, and B $\subseteq$ Y:

A)   g(A) = { g:x | x $\in$ A } and

B) g^{-1}(B) = { x | g:x $\in$ B }.

In addition to the above notation, g(A) and g^{-1}(B) will also be referred

to as the set mapping and inverse set mapping of g respectively.

Proposition 3.2 For all functions g, g:X --> Y, $A_i \subseteq$ X, and $B_i \subseteq$ Y the

following properties hold (10):

1)   If g(ϕ) $\neq$ $\underline{1}$ then g(ϕ) = ϕ 6) g^{-1}(ϕ) = ϕ

2)   g(X) $\subseteq$ Y   7) g^{-1}(Y) = X

3) $A_i \subseteq A_j$ ==> g(A_i) $\subseteq$ g(A_j)   8) $B_i \subseteq B_j$ ==> g^{-1}(B_i) $\subseteq g^{-1}$(B_j)

4)   g(U_iA_i) = U_ig(A_i)   9) g^{-1}(U_iB_i) = U_ig^{-1}(B_i)

5)   g($\cap_iA_i$) $\subseteq \cap_i$g(A_i)   10) g^{-1}($\cap_iB_i$) = $\cap_ig^{-1}$(B)

Proof: Proof that these properties hold is direct by Definition 3.6.

The properties are listed here since they are used frequently in the

proofs of this chapter.

Lemma 3.3.1. For all f in F, for all $f_j{}^c$, j=1,...,p(f), and for all

X,Y $\subseteq$ O the following properties hold:

1) $(f_j{}^c)^{-1}$:O = d($f_j{}^c$)

2) $(f_j{}^c)^{-1}$:Y = $(f_j{}^c)^{-1}$:(Y $\cap$ r($f_j{}^c$))

3) $(f_j{}^c)^{-1}$:Y $\subseteq$ d($f_j{}^c$)

4)   Y $\cap$ X = ϕ ==> (($f_j{}^c)^{-1}$:Y) $\cap$ (($f_j{}^c)^{-1}$:X) = ϕ

Proof: The proofs of these properties are straightforward from

Definition 3.6 and Proposition 3.2.

Lemma 3.3.2. If $d(f1_i{}^c)$, $i=1,\ldots,p(f1)$ partition 0,.... $d(fn_j{}^c)$,

$j=1,\ldots,p(fn)$ partition 0 then:

    1) $d((f1_i{}^c \circledast f2_j{}^c)_1)$, for all (i,j)

         partition $d((f1 \circledast f2)_1)$.

    2) $d([f1_i{}^c,\ldots,fn_j{}^c]_1)$, for all $(i,\ldots,j)$

         partition $d([f1,\ldots,fn]_1)$.

    3) $d([f1_i{}^c,\ldots,fn_j{}^c]_2(Q_1,\ldots Q_n))$, for all $(i,\ldots,j)$, all $(Q_1,\ldots,Q_n)$

         partition $d([f1,\ldots,fn]_2)$.

    4) $d((f1_i{}^c\text{--}>f2_j{}^c;f3)_1)$, for all (i,j)

         partition $d((f1\text{-}>f2;f3)_1)$.

    5) $d((f1_i{}^c\text{--}>f2;f3_k{}^c)_2)$, for all (i,k)

         partition $d((f1\text{--}>f2;f3)_2)$.

    6) $d((f1_i{}^c\text{--}>f2;f3)_3)$, for all i

         partition $d((f1\text{--}>f2;f3)_3)$.

Proof: Proof for cases 1 and 3 are given below. The other cases are
similar and are not given here. This lemma is used for both the basis
and the inductive step of Theorem 3.3.

case 1) Let $d(f1_i{}^c)$, $i=1,\ldots,p(f1)$ partition 0 and $d(f2_j{}^c)$,

$j=1,\ldots,p(f2)$ partition 0. First an equation is derived for the inverse
set mapping.

By Definition 3.6.A, for all $Y \subseteq 0$:

$((f1_i{}^c \circledast f2_j{}^c)_1)^{-1}:Y = \{x \mid (f1_i{}^c \circledast f2_j{}^c)_1:x \in Y\}$

    $= \{x \mid f1_i{}^c(f2_j{}^c:x) \in Y\}$ Df. 3.5

    $= \{x \mid f2_j{}^c:x \in \{y \mid f1_i{}^c:y \in Y\}\}$

    $= (f2_j{}^c)^{-1}:(\{y \mid f1_i{}^c:y \in Y\})$ Df. 3.6.B

    $= (f2_j{}^c)^{-1}:((f1_i{}^c)^{-1}:Y)$ Df. 3.6.B

The following proof shows that $((f1_i{}^C \otimes f2_j{}^C)_1)^{-1}:0$, for all (i,j) are

disjoint. By the initial assumption $(f2_j{}^C)^{-1}:0$, for $j=1,\ldots,p(f2)$ are

disjoint. Then by Lemma 3.3.1.4, $(f2_j{}^C)^{-1}:((f1_i{}^C)^{-1}:0)$ for each j and

for all i are disjoint. By Lemma 3.3.1.3,

$$(f2_j{}^C)^{-1}:((f1_i{}^C)^{-1}:0) \subseteq d(f2_j{}^C), \text{ for each } j, \text{ for all } i.$$

Thus, $(f2_j{}^C)^{-1}:((f1_i{}^C)^{-1}:0)$, for all i,j are disjoint.

Now it is shown that $U_{i,j}(f1_i{}^C \otimes f2_j{}^C)_1)^{-1}:0 = 0$.

Suppose $x \in U_{i,j}((f1_i{}^C \otimes f2_j{}^C)_1)^{-1}:0$.

> $\Longrightarrow x \in d((f1_i{}^C \otimes f2_j{}^C)_1)$ Lemma 3.3.1.1

> $\Longrightarrow x \in d(f2_j{}^C)$ and $(f2_j{}^C:x) \in d(f1_i{}^C)$ Df. 3.6.B

Then, by contradiction $x \in 0$, since $d(f2_j{}^C)=\emptyset \Longrightarrow x \notin d(f2_j{}^C)$.

Thus, $U_{i,j}((f1_i{}^C \otimes f2_j{}^C)_1)^{-1}:0 \subseteq 0$.

Now suppose $x \in 0$. By the initial assumption, $U_j((f2_j{}^C)^{-1}:0) = 0$.

> $\Longrightarrow x \in (f2_j{}^C)^{-1}:0$, for some $j=1..p(f)$.

> $\Longrightarrow f2_j{}^C:x = x'$ for some $x' \in 0$.   Df. 3.6.B.

Then , also by the initial assumption, $U_i(f1_i{}^C)^{-1}:0 = 0$.

> $\Longrightarrow f1_i{}^C:x' \in 0$, for some $i=1,\ldots,p(f1)$.

Then, $f1_i{}^C:(f2_j{}^C:x) \in 0$ for some i,j.

> $\Longrightarrow (f1_i{}^C \otimes f2_j{}^C)_1:x \in 0$.   Df. 3.5.A

> $\Longrightarrow x \in ((f1_i{}^C \otimes f2_j{}^C)_1)^{-1}:0$.   Df. 3.6.B

> $\Longrightarrow x \in U_{i,j}((f1_i{}^C \otimes f2_j{}^C)_1)^{-1}:0$

Thus, $U_{i,j}((f1_1{}^C \otimes f2_j{}^C)_1)^{-1}:0 = 0$.

Therefore, $((f1_i{}^C \otimes f2_j{}^C)_1)^{-1}:0$ for all (i,j) partition $0=d((f1 \otimes f2)_1)$.

Thus, by Lemma 3.3.1.1, $d((f1_i{}^C \otimes f2_j{}^C)_1)$, for all $(i,\ldots,j)$ partition

$d((f1 \longrightarrow f2;f3)_1)$.

case 3) The computational restrictions induced by $[f1,\ldots,fn]_2$ are:

$$Q_1(f1_k{}^c:x=\underline{!}) \&\ldots\& Q_n(fn_m{}^c:x=\underline{!}) \longrightarrow \underline{!}$$

for all (k,m) for all $(Q_1,\ldots,Q_n)$ where $Q_i=$True or $Q_i=$False, $i=1,\ldots,n$

and $Q_i=$True for at least one i. Denote this set of restrictions by $f_j{}^c$,

$j=1,\ldots,p(f)$. By Definition 3.5.B, for each k,m and for each $Q_1,\ldots,Q_n$.

$$d(f_j{}^c) = \{x \mid Q_1(f1_k{}^c:x=\underline{!}) \&\ldots\& Q_n(fn_m{}^c:x=\underline{!}) \}$$

$$\bigcap \{x \mid x \in d(f1_k{}^c),\ldots, x \in d(fn_m{}^c) \}$$

Then $x \in d(f_j{}^c)$

$\implies x \in \{x \mid Q_1(f1_k{}^c:x=\underline{!}) \&\ldots\& Q_n(fn_m{}^c:x=\underline{!}) \}$

$Q_1=$False $\implies {\sim}f1_k{}^c:x=\underline{!} \implies f1_k{}^c:x \neq \underline{!} \implies f1_k{}^c:x \in (O-\underline{!})$

$\implies x \in (f1_k{}^c)^{-1}:(O-\underline{!})$

The same proof holds for $f2,\ldots,fn$.

$\implies x \in (f1_k{}^c)^{-1}:A_1 \bigcap \ldots \bigcap (fn_m{}^c)^{-1}:(A_n)$

where $A_i=\underline{!}$ if $Q_i=$True and $A_i=O-\underline{!}$ if $Q_i=$False.

Consider $f_a{}^c:x$ and $f_b{}^c:x$, $a \neq b$, $a,b=1,\ldots,p(f)$. It must be true that

at least one term in some position i differs in the expressions for

$d(f_a{}^c)$ and $d(f_b{}^c)$. Let these terms be $Q_i(fi_k{}^c:x=\underline{!})$ and $Q'_i(fi_m{}^c:x=\underline{!})$

respectively.

$Q_i \neq Q'_i \& k=i \implies A_i \bigcap A'_i = \emptyset$

$\implies (fi_k{}^c)^{-1}:A_i \bigcap (fi_m{}^c)^{-1}:A'_i = \emptyset$ Lm. 3.3.1.4

$\implies d(f_a{}^c) \bigcap d(f_b{}^c) = \emptyset$

If $k \neq m$, then by the initial assumption about $f_i{}^c$,

$(fi_k{}^c)^{-1}:A_i \bigcap (fi_m{}^c)^{-1}:A_i = \emptyset$, regardless of Q_i,Q'_i.

Thus, if $a \neq b$ then some i'th terms must be disjoint and since $f_a{}^c$, $f_b{}^c$

are arbitrarily selected, $d(f_j{}^c)$, $j=1,\ldots,p(f)$ are disjoint.

Now it is proved that $U_j d(f_j{}^c) = d([f1,\ldots,fn]_2)$. By Definition 3.5.A

$x \in d([f1,\ldots,fn]_2) \implies f1:x=\underline{!} \,!\ldots! \, fn:x=\underline{!}$

==> fi:x=$\underline{i}$ for one or more i, i=1,...,n

==> fi$_j{}^c$:x=$\underline{i}$ for one or more i, and some j=1,...,n

    & ~(fi$_k$:x=$\underline{i}$), for k ≠ j   Df. 3.6.A

==> x ∈ {x| Q$_1$(f1$_i{}^c$:x=$\underline{i}$) &...& Q$_n$(fn$_j{}^c$:x=$\underline{i}$)}

    for some (Q$_1$,...,Q$_n$), some (i,...,j). and some Q$_i$=True.

Thus, d([f1,...,fn]$_2$) ⊆ U$_j$d(f$_j{}^c$).

Now suppose x ∈ U$_j$d(f$_j{}^c$)

    ==> fi$_j{}^c$:x=$\underline{i}$ for at least one i, some j=1,...,p(f$_i$)

    ==> fi:x=$\underline{i}$ for at least one i   Th. 3.1

    ==> f1:x=$\underline{i}$!...! fn:x=$\underline{i}$

Hence, U$_j$d(f$_j{}^c$) ⊆ d([f1,...,fn]$_2$).

Thus, U$_j$d(f$_j{}^c$) = d([f1,...,fn]$_2$).

Therefore, d(f$_j{}^c$), j=1,...,p(f) partition d([f1,...,fn]$_2$).

Theorem 3.3. For any f in F, the domains of the computational

restrictions of f partition O.

Proof: By Proposition 3.1, d(f$_i$), i=1,...,flast partition O. Thus, the

theorem can be proved by showing that the domains of the computational

restrictions induced by each f$_i$ partition d(f$_i$). The possible cases for

f$_i$ are given below.

case 1) For all cases where f is a primitive function, f$_i$ induces only

f$_i{}^c$ and the proof is trivial.

case 2) Other cases must be one of the 6 cases of Lemma 3.3.2. A proof

by induction is given for these cases.

Let S(N) be the statement "If f=G(f1,...,fn) where f is defined by at

most N applications of the definitions of the functional forms in G,

then the domains of the computational restrictions induced by each f$_i$

partition d(fi)."

Basis: If N=1 then f1,...,fn are primitive functions. Then by

Definition 3.4 $f_i = f_i{}^C$, i=1,...,flast and by Proposition 3.1, $d(f_i{}^C)$,

i=1,...,flast partition O. Then by the proofs for cases 1-6 of

Lemma 3.3.2, S(1) is true.

Inductive Step: Suppose S(N) is true for all N $\geq$ 1 and f is defined by

N+1 applications of the definitions of the functional forms in G. Then

f1,...,fn must each be defined by M$\leq$N applications of the definitions of

G. Then S(N) is true for f1,...,fn and by the proofs for cases 1-6 of .

Lemma 3.3.2, the computational restrictions of f partition O.

If S(N) is true for any N $\geq$ 1, then S(N+1) is true. Therefore, by

induction the domains of the computational restrictions of f partition

O.

Methods

In this section methods are shown to construct an equation for the

domain and range of each computational restriction.

Definition 3.7. The following definitions give the domains of the set

mappings of Definition 3.6 for the computational restrictions of f in F.

A) A class of sets $D_1{}^C$ is defined as follows. Let D be a subset of

O, D = D' U D", where D' = D $\cap$ (Atoms U $\underline{\downarrow}$) and D" is the set of all

sequences in D. Then D is in $D_1{}^C$ iff the following properties hold.

1) N $\subseteq$ D' & N $\subseteq$ NUM ==> N=NUM ! N=$\emptyset$.

2) $x = \langle x_1,..,x_j,...,x_n \rangle$ & $x' = \langle x'_1,...,x'_n \rangle$ are in D"

    ==> $x" = \langle x'_1,...x_j,...,x'_n \rangle$ is in D".,

When D has this property it is said to be in closed form and the

set of all sequences of length n in D is denoted by $\langle D_1,....,D_n \rangle$.

$D_1, \ldots, D_n$ are also sets and

$$<x_1, \ldots, x_n> \in D \implies x_i \in D_i, \ i=1, \ldots, n.$$

When the set D contains sequences of lengths $\geq$ n, D is denoted by:

$$U_i <D_1, \ldots, D_k{}^i, \ldots, D_n>$$

$$= <D_1, \ldots, D_k, \ldots, D_n> \cup <D_1, \ldots, D_k, D_k, \ldots, D_n> \cup \ldots$$

3) $D" = <D_1, \ldots, D_n>$ is in $D_1{}^c$, for $n \geq 1$

$\implies D_j$, $j=1, \ldots, n$ is in $D_1{}^c$

4) Only what is defined above is in $D_1{}^c$.

5) The finite unions of $D_1, \ldots, D_n$ in $D_1{}^c$ are denoted $UD_1{}^c$.

In the proofs of Lemmas 3.4.1 3.4.5 and Theorem 3.4, $UD_1{}^c$ is shown

to be the domain of $(f_j{}^c)^{-1}$, for all f and for all $f_j{}^c$,

$j=1, \ldots, p(f)$.

B) Let $D^c \subseteq O$ be the class of sets defined by A) above except that

the limitations of property 2) are relaxed in the definition of

$D \in D^c$ as follows:

2) $N \subseteq D' \ \& \ N \subseteq NUM \implies N=NUM \ ! \ N=\emptyset \ ! \ N=\{x\}$ for some x in NUM.

The finite unions of $D_1, \ldots, D_n$ in D^c are denoted UD^c. In the proofs

of Lemmas 3.4.1 - 3.4.5, Lemmas 3.5.1 - 3.5.4, and Theorem 3.5, UD^c

is shown to be the domain of $f_j{}^c$, for all f and all $f_j{}^c$, $j=1, \ldots, p(f)$

It is clear that $D_1{}^c$ and D^c have similar properties and in

particular that $D_1{}^c$ is contained in D^c. The reason for the restriction

imposed by property 2 of A is that for op $\in \{+,*,sub,div\}$, $(op_i{}^c)^{-1}:D$ is

a relation on (D,X) for each y in D and $X=d(op_i{}^c)=<NUM,NUM>$, and cannot

be given in closed form for the general case of D. The set mapping

$f_j{}^c:D$ is a function for each x in D. Since $D_1{}^c$ is contained in D^c, the

proofs that follow are for the general case of D^c, and $D_1{}^c$ is referred

to only where it is necessary to make a distinction between the two

domains. In particular, Theorem 3.4 shows that UD^c is closed under $(f_j{}^c)^{-1}$, and it is understood that if $op_1{}^c$ occurs in the definition of $f_j{}^c$, that closure is only for $UD_1{}^c$.

The motivation for the above definitions is to give a class of sets which is closed under the set mappings and inverse set mappings of this chapter. The closed form notation is used to simplify examples and as a graphic aid to describe the inverse set mappings of Theorem 3.4. An extension of this closed form notation is given later to describe the set mappings of Theorem 3.5. The separate definitions of D^c and UD^c are required by various proofs that follow. By the above definition it is clear that if $D=<D_1,\ldots,D_n>$ and $D_i=\phi$ for some $i=1,\ldots,n$ then $D=\phi$.

Lemma 3.4.1. The following sets are in UD^c:

1) 0

2) Any subset of Atoms.

3) $r(f_i{}^c)$, $i=1,\ldots,f$last when f is a primitive function in F

4) $d(f_i{}^c)$, $i=1,\ldots,f$last when f is a primitive function in F

   except for the following cases:

   $d(f_1{}^c)$, $d(f_2{}^c)$ for f ∈ R.

Proof: The closed forms of $d(f_i{}^c)$, $r(f_i{}^c)$, $i=1,\ldots,f$last are listed in Appendix C. Proofs are not given for the obvious cases in 1-4. Let f be in F and $Z = 0-\underline{i}$.

case 1)

$0 = $ Atoms U $\underline{i}$ $U_i<Z^i>$

$\quad = $ Atoms U $\underline{i}$ $U_i<($Atoms $U_j<Z^j>)^i>$

Then, clearly properties 1 and 2 of Definition 3.7 hold for 0, and by a simple inductive proof on the number of expansions of $Z=0-\underline{i}$, 0 is in D^c. This same proof shows that all nontrivial cases of 3 and 4 are in UD^c.

case 4) For the exceptions in case 4, f in R, $d(f_1{}^c)$ is listed in Appendix C but cannot be given in closed form. A proof for one case is given to illustrate the nature of these exceptions. Consider

$$d(\geq_1{}^c) = \{x \mid x = <x_1, x_2>, \; x_1, x_2 \in NUM, \; x_1 \geq x_2\}$$

Then $<1,1>$ and $<3,2>$ are in $d(\geq_1{}^c)$ but $<1,2>$ is not. This contradicts property 1 of Definition 3.7 and thus, $d(\geq_1{}^c)$ is not in D^c. The exceptions in case 4 can be expressed in an extension of closed form by showing the relation on $d(f_1{}^c)$ for f in R. This is denoted by $(f_1, <NUM,NUM>)$ for $d(f_1{}^c)$ and $(\sim f_1, <NUM,NUM>)$ for $d(f_2{}^c)$. Clearly $d(f_1{}^c) \cup d(f_2{}^c) = <NUM,NUM>$ for all cases of f given as the exceptions in case 4. This property is used later to include these functions in useful characterizations of f in F.

<u>Lemma 3.4.2.</u> If D, D' are in UD^c then $D \cap D'$ is in UD^c.

<u>Proof:</u> Let D, D' be in UD^c. It is shown that properties 1-3 of Definition 3.7 hold for $D \cap D'$. Consider D,D' in D^c. Let $D = <D_1, \ldots, D_n>$ and $D' = <D'_1, \ldots, D'_n>$.

case a) By property 1 of Definition 3.7, if $N = (D \cap NUM)$ and $N' = (D' \cap NUM)$, then

$$N = \phi \;!\; N' = \phi \implies (NUM \cap (D \cap D')) = \phi$$

and

$$N = NUM \;\&\; N' = NUM \implies (NUM \cap (D \cap D')) = NUM.$$

Thus, property 1 holds for $D \cap D'$.

case b) Let $x = <x_1, \ldots, x_j, \ldots, x_n> \;\&\; x' = <x'_1, \ldots, x'_n> \in (D \cap D')$.

$\implies x'' = <x'_1, \ldots, x_j, \ldots, x'_n> \in D \;\&\; x'' \in D'.$ 　　　　Df.3.7

$\implies x'' \in (D \cap D')$

Thus, property 2 of Definition 3.7 holds for $D \cap D'$.

case c) To prove property 3 it must be shown that

$(D \cap D') = D'' = <D''_1, \ldots, D''_n>$ for some $D''_1, \ldots, D''_n$ in D^c.

$D \cap D' = \{x \mid x \in D \text{ and } x \in D'\}$

    $= \{x \mid x_i \in D_i \ \& \ x_i \in D'_i, \ i=1, \ldots, n\}$ Df. 3.7.2

    $= \{x \mid x_i \in (D_i \cap D'_i), \ i=1, \ldots, n)\}$

    $= <(D_1 \cap D'_1), \ldots, (D_n \cap D'_n)>$ Df. 3.7.2

Then since D_i, D'_i, $i=1, \ldots, n$ are in D^c by property 3 of Definition 3.7,

properties 1 and 2 of Definition 3.7 hold for D_i, D'_i, $i=1, \ldots, n$. Then

by induction on the number of applications of this expansion of $D \cap D'$,

property 3 holds for $D \cap D'$.

Thus, from a-c, $D \cap D' \in D^c$, $D = <D_1, \ldots, D_n>$ and $D' = <D'_1, \ldots, D'_n>$.

Now consider

    $D = U_i <D_1, \ldots, D_k^i, \ldots, D_n>$ and

    $D' = U_j <D'_1, \ldots, D'_m{}^j, \ldots, D'_n>$.

Then for each $n'' \geq \max(n, n')$, the intersection of sequences of length n''

in

$D \cap D'$ is either empty or is in D^c by the above proof for 1.

Thus, $D \cap D' \in D^c$.

Now suppose $D = U_i D_i$, $i=1, \ldots, n$ and $D' = U_j D_j$, $j=1, \ldots, n'$ in UD^c. Then

by DeMorgan's Laws

$D \cap D' = D \cap (D'_1 \cup \ldots \cup D'_n)$

    $= (D \cap D'_1) \cup \ldots \cup (D \cap D'_n)$

    $= ((D_1 \cap D'_1) \cup \ldots \cup (D_n \cap D'_1)) \cup \ldots$

        $\cup ((D_1 \cap D'_n) \cup \ldots \cup (D_n \cap \ldots \cap D'_n))$

Then the expression given is a finite union of intersections of sets in

D^c which by property 5 of Definition 3.7 is in UD^c.

Therefore, if D, D' are in UD^c, $D \cap D'$ is in UD^c.

<u>Lemma 3.4.3.</u> If D is in UD^c and f is a primitive function if F, then

$(f_i{}^c)^{-1}$: D is in UD^c,

except for $f_i{}^c$, 1=1,2, f $\in$ R

<u>Proof:</u> Appendix D lists $(f_i{}^c)^{-1}$:D, D $\subseteq$ $r(f_i{}^c)$, for all primitive

functions in F. For each f that is not an exception above, the equation

for $(f_i{}^c)^{-1}$:D is given in closed form. Thus, UD^c is closed under the

equations given in Appendix D. By Lemma 3.3.1.2

$(f_i{}^c)^{-1}$:D = $(f_i{}^c)^{-1}$:(D $\bigcap$ $r(f_i{}^c)$)

which is in D^c by Lemma 3.3.2 and Lemma 3.3.3. Then it is only

necessary to show that the equations given for $(f_i{}^c)^{-1}$:D in Appendix D

are correct. One example case is proved below. The other cases are

similar.

For X,Y $\in$ O, and D $\in$ UD^c:

From Definition 3.6:

1) $(tl_2{}^c)^{-1}$:Y = $\{x \mid tl_2{}^c$:x $\in$ Y$\}$

From Appendix D:

2) $(tl_2{}^c)^{-1}$:D $\equiv$ D $\subseteq$ $U_i<Z^i>$ --> $U_i<Z,D_1,\ldots,D_k{}^i,\ldots,D_n>$

These two equations are equivalent for all D $\subseteq$ $r(f_i{}^c)$ by the following

proof.

$(tl_2{}^c)^{-1}$:D = $(tl_2{}^c)^{-1}$: $U_i<D_1,\ldots,D_k{}^i,\ldots,D_n>$

$= U_i((tl_2{}^c)^{-1}:<D_1,\ldots,D_k{}^i,\ldots,D_n>)$ Pr. 3.2.9

$= U_i((tl_2{}^c)^{-1}:\{y \mid y=<y_1,\ldots,y_k{}^i,\ldots,y_n>,$

for $y_j \in D_j$, j=1,$\ldots$,n)   Df. 3.7.2

$= U_i(\{x \mid x=<x_1,y_1,\ldots,y_k{}^i,\ldots,y_n>,$

$x_1 \in O-\underline{i}$, $y_j \in D_j$, j=1,$\ldots$,n]$\}$ App. A

$= U_i<Z,D_1,\ldots,D_k{}^i,\ldots,D_n>$ Df. 3.7.2

Thus, the inverse set mapping for $tl_2{}^c$ in Appendix D is correct.

Lemma 3.4.4. For all $f1, f2$ in F, for all $f1_i^c$, $i=1,\ldots,p(f1)$, $f2_j^c$,

$j=1,\ldots,p(f2)$, and for all $Y \subseteq O$:

$$((f1_i^c \circ f2_j^c)_1^c)^{-1}:Y = (f2_j^c)^{-1}:((f1_i^c)^{-1}:Y)$$

Proof: The proof for this Lemma is given by the derivation of the

equation in the proof of Lemma 3.3.2.1. An example is given below to

illustrate a computation of this inverse set mapping.

Example 3.3. Let $f_i = (+\circ tI)_1$ and $Z = O-\underline{i}$.

1) $((+_1^c \circ tI_1^c)_1)^{-1}:O$

$$= (tI_1^c)^{-1}:((+_1^c)^{-1}:O) \qquad\qquad \text{Lm. 3.4.4}$$

$$= (tI_1^c)^{-1}:(d(+_1^c)) \qquad\qquad \text{Lm. 3.3.1.1}$$

$$= (tI_1^c)^{-1}:(d(+_1^c) \cap r(tI_1^c)) \qquad\qquad \text{Lm. 3.3.1.2}$$

$$= (tI_1^c)^{-1}: \phi \qquad\qquad \text{App. D}$$

$$= \phi \qquad\qquad \text{Pr. 3.2.6}$$

2) $((+_1^c \circ tI_2^c)_1)^{-1}:O)$

$$= (tI_2^c)^{-1}:(d(+_1^c) \cap r(tI_2^c)) \qquad\qquad \text{Ex. 3.3.1}$$

$$= <Z, NUM, NUM> \qquad\qquad \text{App. D,E}$$

By Lemma 3.3.1.1 for all (i,j),

$$((+_i^c \circ tI_j^c)_1)^{-1}:O = d((+_i^c \circ tI_j^c)_1^c).$$

Then by similar methods:

$$d((+_1^c \circ tI_3^c)_1) = \phi$$

$$d((+_2^c \circ tI_1^c)_1) = <Z>$$

$$d((+_2^c \circ tI_2^c)_1) = <Z,Z> \cup <Z,Z-NUM,Z-NUM>$$

$$\cup <Z,Z,Z-NUM> \cup <Z,Z-NUM,Z>$$

$$\cup_i <Z,Z,Z,Z^i>$$

$$d((+_2^c \circ tI_3^c)_1) = ATOMS \cup \underline{i}$$

Definition 3.8. The following notation is defined for all f in F and for all $f_j{}^c$, $j=1,\ldots,p(f)$:

$$U_R f_j{}^c$$

denotes all $f_j{}^c$ such that $r_1{}^c$ or $r_2{}^c$, $r \in R$ does not occur in the definition of $f_j{}^c$.

This notation and natural extensions such as $U_R d(f_j{}^c)$ will be used extensively later in this chapter and in the following chapter. The definition is given here because UD^c is not closed under the inverse set mappings for $r_1{}^c$, $r_2{}^c$ shown in Lemma 3.4.4. Thus, the equation given in the following lemma and the constructed equation for $d(f_j{}^c)$ of Theorem 3.4 are not valid for these cases. A method for deriving $d(f_j{}^c)$, $f_j{}^c \in U_R f_j{}^c$ can only be given following Theorem 3.5.

Lemma 3.4.5. For all f, for all $f1_i{}^c,\ldots,fn_j{}^c$ not in $U_R f1_k{}^c,\ldots,U_R fn_k{}^c$, and for all $D \in UD^c$, $D=<D_1,\ldots,D_n>$

1) $([f1_i{}^c,\ldots,fn_j{}^c]_1)^{-1}:D = (f1_i{}^c)^{-1}:(D_1-\underline{i}) \cap \ldots \cap (fn_j{}^c)^{-1}:(D_n-\underline{i})$

2) $([f1_i{}^c,\ldots,fn_j{}^c]_2(Q_1,\ldots Q_n))^{-1}:D = (f1_i{}^c)^{-1}:A_1 \cap \ldots \cap (fn_j{}^c)^{-1}:A_n$.

 where $A_i=\underline{i}$ if $Q_1=$True, $A_i=0-\underline{i}$ if $Q_i=$False, $i=1,\ldots,n$.

Proof: A proof for case 1 is given below. Case 2 is proved in Lemma 3.3.2.2.

case 1) By Definition 3.6.2

$([f1_i{}^c,\ldots,fn_j{}^c]_1)^{-1}:D = \{x \mid [f1_i{}^c,\ldots,fn_j{}^c]_1:x \in D\}$

$\quad = \{x \mid \sim f1_i{}^c:x=\underline{i} \&\ldots\& \sim fn_j{}^c:x=\underline{i}$

$\qquad \& <f1_i{}^c:x,\ldots,fn_j{}^c:x> \in D\}$ Df. 3.5.1

$\quad = (f1_i{}^c)^{-1}:(0-\underline{i}) \cap \ldots \cap (fn_j{}^c)^{-1}:(0-\underline{i})$

$\qquad \cap \{x \mid f1_i{}^c:x=y_1,\ldots,fn_j{}^c:x=y_n,$ (1)

$\qquad \text{for some } y=<y_1,\ldots,y_n> \in D\}$ Df. 3.6.2

Now let A = $(f1_i{}^c)^{-1}:D_1 \cap ... \cap (fn_j{}^c)^{-1}:D_n$ and

$\quad$ B = $\{x \mid f1_i{}^c:x=y_1,...,fn_j{}^c:x=y_n, y \in D\}$.

Then

x $\in$ B $\Longrightarrow$ x $\in$ $(f1_i{}^c)^{-1}:\{y_1\}\&...$

$\qquad$ & x $\in$ $(fn_j{}^c)^{-1}:\{y_n\}$ for some y in D

$\quad \Longrightarrow$ x $\in$ $(f1_i{}^c)^{-1}:D_1$ &...& x $\in$ $(fn_j{}^c)^{-1}:D_n$ $\qquad$ Df. 3.7

Thus, B $\subseteq$ A.

Suppose x $\in$ A

$\quad \Longrightarrow$ x $\in$ $(f1^c)^{-1}:\{y_1\}$ &..& x $\in$ $(fn_j{}^c)^{-1}:\{y_n\}$, $y_i \in D_i$

$\quad \Longrightarrow$ x $\in$ $(f1_i{}^c)^{-1}:\{y_1\}$, for some y=<$y_1,...,y_n$> $\in$ D

$\qquad$ ..&.. x $\in$ $(fn_j{}^c)^{-1}:\{z_1\}$ for some z=<$z_1,..z_n$> $\in$ D $\quad$ Df. 3.7

$\quad \Longrightarrow$ <$y_1,...,z_n$> $\in$ D

$\quad \Longrightarrow$ x $\in$ $(f1_i{}^c)^{-1}:\{y_1\}$ &...& x $\in$ $(fn_j{}^c)^{-1}:\{y_n\}$,

$\qquad$ for <$y_1,...,z_n$> in D.

$\quad \Longrightarrow$ x $\in$ B $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ Df. 3.6.B

Thus, A $\subseteq$ B

Therefore, A = B.

By substitution of A for B in equation (1) above:

$([f1_i{}^c,...,fn_j{}^c]_1)^{-1}:D$ = $((f1_i{}^c)^{-1}:D_1 \cap ... \cap (fn_j{}^c)^{-1}:D_n))$

$\qquad\qquad\qquad \cap ((f1_i{}^c)^{-1}:0\underline{-1} \cap ... \cap (fn_j{}^c)^{-1}:0\underline{-1})$

$\quad$ = $((f1_i{}^c)^{-1}:(D_1 \cap 0\underline{-1}) \cap ... \cap (fn_j{}^c)^{-1}:(D_n \cap 0\underline{-1})$ $\qquad$ Pr. 3.2.9

$\quad$ = $((f1_i{}^c)^{-1}:(D_1\underline{-1}) \cap ... \cap (fn_j{}^c)^{-1}:(D_n\underline{-1})$

Thus, the equation for case 1 is derived for D $\in$ UD^c.

Example 3.4.   Let $Z = 0-\underline{!}$

1) a) By Lemma 3.3.3.1, $d([s_1{}^c, t|_1]_1)$

     $= (([s_1{}^c, t|_1{}^c]_1)^{-1} : 0$

     $= (s_1{}^c)^{-1} : Z \cap (t|_1{}^c)^{-1} : Z$ Lm. 3.4.5

     $= (s_1{}^c)^{-1} : (Z \cap r(s_1{}^c))$

        $\cap (t|_1{}^c)^{-1} : (Z \cap r(t|_1{}^c))$ Lm. 3.3.1.2

     $= (s_1{}^c)^{-1} : Z \cap (t|_1{}^c)^{-1} : <>$ App. C

     $= U_| <Z^|> \cap <Z>$ App. D

     $= <Z>$

   b) By similar methods,

     $d([s_1{}^c, t|_2{}^c]_1) = U_| <Z, Z^|>$

     $d([s_1{}^c, t|_3{}^c]_1) = \phi$

     $d([s_2{}^c, t|_j{}^c]_1) = \phi, \ j=1, \ldots, 3$

   Thus, $d([s_i{}^c, t|_j{}^c]_1)$, $i=1,2$, $j=1, \ldots, 3$

   partition $d([f1, \ldots, fn]_1)$

2) By Lemma 3.4.5.2

   a) $d([s_2{}^c, t|_3{}^c]_2(T, T))$

     $= (s_2{}^c)^{-1} : (0-\underline{!}) \cap (t|_3{}^c)^{-1} : (0-\underline{!})$

     $= (s_2{}^c)^{-1} : ((0-\underline{!}) \cap r(s_2{}^c))$

        $\cap (t|_3{}^c)^{-1} : ((0-\underline{!}) \cap r(t|_3{}^c))$

     $= (s_2{}^c)^{-1} : \underline{!} \cap (t|_3{}^c)^{-1} : \underline{!}$

     $= (\text{Atoms} \cup \underline{!}) \cap (\text{Atoms} \cup \underline{!})$

     $= \text{Atoms} \cup \underline{!}$

   b) For all other cases of (i, j), (Q_1, Q_2)

     $d([s_1{}^c, t|_j{}^c]_2(Q_1, Q_2)) = \phi$

   Thus, $d([s_i{}^c, t|_j{}^c]_k)$, for all (i, j, k) partition 0.

Lemma 3.4.6. For all f1,f2,f3 in F, and for all $f1_i{}^c$, $f2_j{}^c$, and $f3_k{}^c$ not in $U_R f1^c$, $U_R f2^c$, $U_R f3^c$ respectively, and for all $Y \subseteq O$:

1) $((f1_i{}^c \text{-->} f2_j{}^c; f3)_1)^{-1}:Y = (f1_i{}^c)^{-1}:\{T\} \cap (f2_j{}^c)^{-1}:Y$

2) $((f1_i{}^c \text{-->} f2; f3_k{}^c)_2)^{-1}:Y = (f1_i{}^c)^{-1}:\{F\} \cap (f3_k{}^c)^{-1}:Y$

3) $((f1_i{}^c \text{-->} f2; f3)_3)^{-1}:Y = (f1_i{}^c)^{-1}:(O-\{T,F\})$

Proof: Proofs are straightforward from Definition 3.5.A and Definition 3.6.2 and only case 1 is given here. The other cases are similarly proved.

case 1) By Definition 3.6.2 and Definition 3.5.A

$((f1_i{}^c; f2_j{}^c; f3)_1)^{-1}:Y = \{x \mid f1_i{}^c:x=\{T\} \ \& \ f2_j{}^c:x \in Y\}$

$= \{x \mid f1_i{}^c:x=\{T\}\} \cap \{x \mid x \in f2_j{}^c:x=Y\}$

$= (f1_i{}^c)^{-1}:\{T\} \cap (f2_j{}^c)^{-1}:Y$ Df. 3.6.2

Example 3.5. Let $Z=O-\underline{\mathfrak{l}}$, $f = (atom \text{-->} id; tl)$

1) a) By Lemma 3.3.1.1, $d((atom_1{}^c \text{-->} id_1{}^c; tl)_1)$

     $= ((atom_1{}^c \text{-->} id_1{}^c; tl)_1)^{-1}:O$

     $= (atom_1{}^c)^{-1}:T \cap (id_1{}^c)^{-1}:O$ Lm. 3.4.6.1

     $= Atoms \cap O$ App. D.

     $= Atoms$

   b) By similar methods

     $d((atom_1{}^c \text{-->} id_2{}^c; tl)_1) = \phi$

     $d((atom_i{}^c \text{-->} id_j{}^c; tl)_1) = \phi$, i=2,3, j=1,2.

   Thus, $d((atom_i{}^c \text{-->} id_j{}^c; tl)_1)$, i=1..3, j=1,2

   partition $d((atom \text{-->} id; tl)_1)$

2) a) $d((atom_2{}^c \text{-->} id; tl_1{}^c)_2)$

     $= (atom_2{}^c)^{-1}:F \cap (tl_1{}^c)^{-1}:O$

     $= U_i <Z^i> \cap <Z>$

     $= <Z>$

By similar methods

b) $d((atom_2{}^C-->id;tl_2{}^C)_2)) = U_i<Z,Z^i>$

c) For all other cases of (i,j)

$d((atom_i{}^C-->id;tl_j)_2) = \emptyset$

3) a) $d((atom_3{}^C-->id;tl)_3)$

$= (atom_3{}^C)^{-1}:(0-\{T,F\}$

$= (atom_3{}^C)^{-1}:((0-\{T,F\} \cap r(atom_3{}^C))$

$= (atom_3{}^C)^{-1}: \underline{\bot}$

$= \underline{\bot}$

Thus, $d((atom_i{}^C-->id_j{}^C;tl_k{}^C)_i)$, for all (i,j,k,l) partition 0.

In the following theorem the set of equations given in Lemmas 3.4.4 - 3.4.6 are used to construct an equation for $(f_j{}^C)^{-1}:D$ for any f in F, any $f_j{}^C$ not in $U_Rf_i{}^C$, and any D in UD^C. The set UD^C is shown to be closed under all equations constructed. The special case of D = 0 illustrates the derivation of $d(f_j{}^C)$.

<u>Theorem 3.4.</u> For all f in F, for all $f_j{}^C$ not in $U_Rf_i{}^C$, and for all D G UD^C, $(f_j{}^C)^{-1}:D$ is in UD^C.

<u>Proof:</u> It is noted that for $f_j{}^C$ where op G $\{+,*,sub,div\}$ that for D in $UD^C{}_1$, $(f_j{}^C)^{-1}:D$ G $UD^C{}_1$ and hence is in UD^C. Also, for the special case of D=0, by Lemma 3.3.1.1, $(f_j{}^C)^{-1}:D = d(f_j{}^C)$. By Proposition 3.2.4, $(f_j{}^C)^{-1}:U_iD_i = U_i((f_j{}^C)^{-1}:D_i)$ and therefore only the case of D in D^C needs additional proof.

case 1) If f is a primitive function in F then $(f_j{}^C)^{-1}:D$ is given in Appendix D and is in UD^C by Lemma 3.4.3.

case 2) If f=G(f1,...,fn) then for each $f_j{}^C$ an equation for $(f_j{}^C)^{-1}:D$ can be constructed by recursive expansion using the equations of Lemmas 3.4.4 - 3.4.6 and the inverse set mapping equations for primitive

functions given in Appendix E. Since $f_i \neq f$, $i=1,\ldots,n$ the equation has

a finite number of terms which are one of the following:

   i) $(f2_j{}^c)^{-1}:((f1_i{}^c)^{-1}:D')$

   ii) $(f1_i{}^c)^{-1}:D' \cap \ldots \cap (fn_j{}^c)^{-1}:D'$

   iii) $(f1_i{}^c)^{-1}:D' \cap (f2_j{}^c)^{-1}:D'$

Where D' is $\underline{i}$, $O-\underline{i}$, $O-\{T,F\}$, or $D_i-\underline{i}$ for some $D'=<D_1,\ldots,D_n>$. By Lemma

3.4.3 and Definition 3.7, the sets D', $D_1,\ldots,D_n$ are all in UD^c. A

proof by induction is given that if $f=G(f1,\ldots,fn)$, then $(f_j{}^c)^{-1}:D \in UD^c$

for all $f_j{}^c$, $j=1,\ldots,p(f)$.

Let $S(N)$ be the statement that "If f is defined by at most N

applications of the definitions of the functional forms in G, then

$(f_j{}^c)^{-1}:D \in UD^c$."

Basis: If $N=1$ then $f1,\ldots,fn$ are primitive functions in f.

case i) By Lemma 3.4.3 $(f1_i{}^c)^{-1}:D'=D''$ and $(f2_j{}^c)^{-1}:(D'')$ are in UD^c.

case ii) By Lemma 3.4.3 and Lemma 3.4.2 the reduction of

   $(f1_i{}^c)^{-1}:D' \cap \ldots \cap (fn_j{}^c)^{-1}:D'$

gives a set in UD^c.

case iii) By Lemma 3.4.3 and Lemma 3.4.2.,

   $(f1_i{}^c)^{-1}:D' \cap (f2_j{}^c)^{-1}:D'$

reduces to a set in UD^c.

Thus, $S(1)$ is true.

Inductive step: Suppose $S(N)$ is true for any $N\geq1$. Then if f is defined

by $N+1$ applications of the definitions of the functional forms in G,

$f1,\ldots,fn$ are each defined by some $M\leq N$ applications of the definitions

of the functional forms in G. Then $S(N)$ is true for $f1,\ldots,fn$.

case i) Since $S(N)$ is true for $f1$ and $f2$, then

   $(f1_i{}^c)^{-1}:D'$ and $(f2_j{}^c)^{-1}:((f1_i{}^c)^{-1}:D'))$

are in UD^c.

case ii) Since S(N) is true for f1,...fn then

$$(f1_i{}^c)^{-1}:D',\ldots, (fn_j{}^c)^{-1}:D'$$

are in UD^c and by Lemma 3.4.2

$$(f1_i{}^c)^{-1}:D' \cap \ldots \cap (fn_j{}^c)^{-1}:D'$$

is in UD^c.

case iii) Since S(N) is true for f1 and f2 then $(f1_i{}^c)^{-1}:D'$ and

$(f2_j{}^c)^{-1}:D'$ are in UD^c. Then by Lemma 3.4.2

$$(f1_i{}^c)^{-1}:D' \cap (f2_j{}^c)^{-1}:D'$$

is in UD^c.

Then, S(N) is true for all $N \geq 1$.

Therefore, by induction $(f_j{}^c)^{-1}:D \in UD^c$.

Theorem 3.4 shows the construction of an equation for $(f_j{}^c)^{-1}:D$, for all f, all $f_j{}^c$ not in $U_R f_i{}^c$, and all $D \in UD^c$. Solving the equation for $(f_j{}^c)^{-1}:0$ gives $d(f_j{}^c) \in UD^c$. Lemmas 3.5.1 – 3.5.4 and Theorem 3.5 show that by similar methods an equation can be constructed for $f_j{}^c:D$, such that if $D \in UD^c$, $f_j{}^c:D \in UD^c$. Thus, $r(f_j{}^c) \in UD^c$ can be derived by solving the equation for $f_j{}^c:d(f_j{}^c)$.

The differences between the two set mappings of Definition 3.6 are illustrated by the following definitions and observations. The primary difference between the two set mappings is that $(f_j{}^c)^{-1}:D$ maps each y in D to $\{x| f_j{}^c:x=y\}$, while $f_j{}^c:D$ maps each x in D to one y in $f_j{}^c:D$. The significance of this difference is that each y in $f_j{}^c:D$ is bound to some x in D by $f_j{}^c$. The following definitions and examples illustrate this property of the set mapping of Definition 3.6.A.

For each D in UD^c, every $x \in D$ has some $n \geq 1$ component objects each of which has a value attribute and a position attribute. The closed form of D given in Definition 3.7 denotes these attributes for

all x in D, but does not show the binding of $f_j{}^c{:}D$ to D.

Consider $f_j{}^c{:}x = id_1{}^c{:}x$ and $D{=}d(id_1{}^c){=}O$. Then

$f_j{}^c{:}D = \{f_j{}^c{:}x|\ x\ \in\ O\}$ .... Df. 3.6

$= \{x|\ id_1{}^c{:}x|\ x\ \in\ O\}$ .... Df. 3.5

$= \{x|\ x\ \in\ O\}$ .... (1) .... App. D

$= O$ .... (2)

Equation 2 gives $r(f_j{}^c)$ but does not show the binding of each y in $r(id_1{}^c)$ to some x in $d(id_1{}^c)$. Applications in this chapter and in the next chapter require this binding information.

<u>Definition 3.9.</u> For x, f:x in O the following attributes are defined:

1) Let the name attribute of x in O be the name of the position of x as follows:

a) If x $\in$ O and x is not a component object of x' in O then name(x) = X.

b) If x $\in$ O and x is the i'th object of a sequence, x', in O, then name(x) = name(parent(x)).i.

2) Let the bound value attribute of x in O be defined as:

1) If x is in O then bound value(x) = name(x).

2) For all f in F and x in O, the bound value attribute of

f:x = f_i:(bound value(x)), such that $P_i(x){=}True$.

In the examples that follow, name(x) will be denoted in parenthesis to the right of x. Bound value(f:x) is also be shown to the right of f:x in parenthesis.

<u>Example 3.7.</u>

1) The notation for the name attribute of x in O is shown:

if x = <<4,...,last>,T>

then x = <<4(X.1.1),...,last(X.1.last)>(X.1),T(X.2)>(X)

2) The notation for the bound value attribute of $f:x$ in O is shown:

$ses: <<A(X.1.1)>(X.1),T(X.2)>(X)$

$\quad = s:(s: <<A(X.1.1)>(X.1),T(X.2)>(X))$

$\quad = s: <A(X.1.1)>(X.1)$

$\quad = A(X.1.1)$

3) The value attribute of a sequence x in O is the length of x

$tl: <12(X.1), F(X.2)>(X) = <F(X.2)>(|X|-1)$

4) The notation described above extends naturally to sets in closed form to show the binding of $f_j{}^c:D$ to D.

$+_1{}^c: <NUM(X.1), NUM(X.1)>(X) = NUM(X.1 + X.2)$

$tl_2{}^c: <NUM(X.1), NUM(X.2)>(X) = <NUM(X.2)>(|X|-1)$

Another difference between the two set mappings of Definition 3.6 is that $f_j{}^c:D$ is defined only for $D \subseteq d(f_j{}^c)$. The following definition is given so that in Theorem 3.5 an equation can be constructed for $f_j{}^c:D$ which is defined for all $D \subseteq UD^c$. In particular, this result is used later to extend the characterization of f in F by showing that $d(f_j{}^c)$ can be derived for $f_j{}^c$ in $U_R f_j{}^c$.

Definition 3.10. Definition 3.6.A is extended as follows. For all f in F, for all $f_j{}^c$, $j=1,\ldots,p(f)$, and for all $D \subseteq UD^c$

$\quad f_j{}^c:D = f_j{}^c:(D \cap d(f_j{}^c))$.

If $D \subseteq d(f_j{}^c)$ then Definition 3.6.A clearly applies. If $d(f_j{}^c) \subseteq D$ then there are no contradictions of Definition 3.6.A implied by this extension since $f_j{}^c:(D-d(f_j{}^c)) = \emptyset$. Equations are given for $f_j{}^c:D$, $D \subseteq UD^c$, in the lemmas and Theorem 3.5 that follow. For the special case of $D \subseteq d(f_j{}^c)$ the extended definition is not necessary. It is given to show that the equations are still correct when $D \not\subseteq d(f_j{}^c)$.

<u>Lemma 3.5.1.</u> If f is a primitive function in F and D is in UD^c then

   $f_1{}^c$:D is in UD^c.

<u>Proof:</u> Either f $\in$ R or f $\notin$ R.

case 1) If f $\in$ R, then $f_j{}^c$:D = $f_j{}^c$:(D$\cap$d($f_j{}^c$)) by Definition 3.10. Then

by property 2 of Definition 3.7.B, D=<D_1,D_2> where D_1={x} or D_1=NUM and

D_2={x'} or D_2=NUM, for x,x' in 0. In any of these cases $f_1{}^c$:D={T} and

$f_2{}^c$:D={F} and the intersection need not be evaluated to determine that

$f_j{}^c$:D is in UD^c.

case 2) For f $\notin$ R, by Proposition 3.2.4, $f_1{}^c$:$U_i D_i$ = $U_i f_1{}^c$:D_i.

Therefore, only D $\in$ D^c needs additional proof. Assume that the

equations for $f_1{}^c$:D, D $\subseteq$ d($f_1{}^c$) given in Appendix E are correct. The

equations are all given in closed form and by Definition 3.7 it is clear

that if D $\in$ D^c, D $\subseteq$ d($f_1{}^c$), then $f_1{}^c$:D $\in$ D^c. By Lemma 3.4.2,

D $\cap$ d($f_1{}^c$) is in UD^c and thus the only additional proof required is to

show that the equations of Appendix E are correct. One case is proved

here. Proofs of the other cases are similar.

Consider

   $s_1{}^c$:x $\equiv$ x=<x_1,...,x_n>,n$\geq$1 --> x_1.

If D $\in$ D^c, D $\subseteq$ d($s_1{}^c$) then D=<D_1,...,D_n> or D=U_i<D_1,...,$D_k{}^i$,...,D_n>.

Only the second case is proved. The case of D= <D_1,...,D_n> is similar.

By Definition 3.6.A, $s_1{}^c$:D = {$s_1{}^c$:x $|$ x $\in$ D}

   = {$s_1{}^c$:x $|$ x=<x_1,...,$x_k{}^i$,...,x_n>, i$\geq$1, x_j $\in$ D_j, j=1,...,n+i}

                                                      Df. 3.7

   = {x_1 $|$ x_1 $\in$ D_1(X.1)}   Df. 3.9

   = D_1(X.1)   Df. 3.7

Thus, the set mapping equation in Appendix E is correct for all

D $\subseteq$ d($s_1{}^c$).

<u>Lemma 3.5.2</u> For all f1, f2 in F, $f1_i{}^C$, $f2_j{}^C$ and for all $D \in UD^C$

$$(f1_i{}^C \circledast f2_j{}^C)_1 : D = f1_i{}^C : ((f2_j{}^C : (D | | d(f2_j{}^C))) \cap d(f1_i{}^C))$$

<u>Proof:</u> Let $D \in UD^C$, and $f_k{}^C$ denote $(f1_i{}^C \circledast f2_j{}^C)_1$. By Definition 3.6.A

$(f1_i{}^C \circledast f2_j{}^C)_1 : D = \{(f1_i{}^C \circledast f2_j{}^C)_1 : x \mid x \in D \cap d(f_k{}^C)\}$

$= \{f1_i{}^C : (f2_j{}^C : x) \mid x \in D \cap d(f_k{}^C)\}$ Df. 3.5

$= \{f1_i{}^C : x' \mid x' \in \{f2_j{}^C : x \mid x \in D \cap d(f_k{}^C)\}\}$

$= \{f1_i{}^C : x' \mid x' \in f2_j{}^C : D \cap d(f_k{}^C)\}$ Df. 3.6.A

$= f1_i{}^C : (f2_j{}^C : D \cap d(f_k{}^C))$ Df. 3.6.A

$= f1_i{}^C : (f2_j{}^C : (D \cap d(f_k{}^C) \cap d(f2_j{}^C))) \cap d(f1_i{}^C)$ Df. 3.10

$= f1_i{}^C : ((f2_j{}^C : (D \cap d(f2_j{}^C))) \cap d(f1_i{}^C))$ Pr. 3.2.8

Thus, the given equation is correct.

In the example that follows the equation for $f_j{}^C : D$ given above is used to derive $f_j{}^C : d(f_j{}^C) = r(f_j{}^C)$. A proof similar to the proof above can be used to show that

$$(f1_i{}^C \circledast f2_j{}^C)_1 : D = f1_i{}^C : (f2_j{}^C : D)$$

when $D \subseteq d(f_j{}^C)$. The only difference in the proofs is the substitution of Definition 3.6.A for Definition 3.10 in the proof above.

<u>Example 3.7.</u> The example is a continuation of the computations given in Example 3.3. For all cases of $d(f_j{}^C) \neq \emptyset$, $r(f_j{}^C)$ is derived from the set mapping equation given in the above lemma for $f_j{}^C : D$.

     a) By Definition 3.5.B

     $r((+_1{}^C \circledast t I_2{}^C)_1)$

         $= (+_1{}^C \circledast t I_2)_1{}^C : d((+_1{}^C \circledast t I_2{}^C)_1)$

         $= +_1{}^C : (t I_2{}^C : <Z(X.1), NUM(X.2), NUM(X.3)>(X) \cap d(t I_2{}^C))$ Lm. 3.5.2

         $= +_1{}^C : (<NUM(X.2), NUM(X.3)>(|X|-1) \cap d(+_1{}^C))$ App. E

         $= NUM(X.2 + X.3)$ App. E

b) From example a above

$r((+2^C \oplus t|1_1^C)_1) = +2^C:(t|1_1^C:(<Z(X.1)>(X) \cap d(t|1_1^C)))$

$= +2^C:(<> \cap d(+2^C))$ App. E

$= +2^C: <>$

$= \underline{\perp}$

c) $r((+2^C \oplus t|2_2^C)_1)$

$= +2^C:(t|2_2^C: d((+2^C \oplus t|2_2^C)_1 \cap d(t|2_2^C))$

$= +2^C:(t|2_2^C: <Z,Z> \cup <Z,Z-NUM,Z-NUM> \cup_| <Z,Z,Z,Z^|>))$

$= +2^C:(<Z> \cup <Z-NUM,Z-NUM> \cup <Z,Z-NUM>$

         $\cup <Z-NUM,Z> \cup_| <Z,Z,Z^|>) \cap d(+2^C))$

$= \underline{\perp}$

d) $r((+2^C \oplus t|3_3^C)_1)$

$= +2^C:(t|3_3^C:(Atoms \cup \underline{\perp} \cap d(t|3_3^C))) \cap d(+2^C)$

$= +2^C:(\underline{\perp} \cap d(+2^C))$

$= \underline{\perp}$

Lemma 3.5.3. For all f1,...,fn in F, for all $f1_|^C,...,fn_J^C$, and for all

$D \in UD^C$,

    $[f1_|^C,...,fn_J^C]_1:D$

        $= <f1_|^C:(D \cap d(f1_|^C) \cap D'),...,fn_J^C:(D \cap d(fn_J^C) \cap D')>$

         where $D' = (f1_|^C)^{-1}:(O-\underline{\perp}) \cap ... \cap (fn_J^C)^{-1}:(O-\underline{\perp})$

Proof: Let $f_K^C = [f1_|^C,...,fn_J^C]_1$. By Definition 3.10

$[f1_|^C,...,fn_J^C]_1:D = \{[f1_|^C,...,fn_J^C]_1:x| \ x \in D \cap d(f_K^C))\}$

    $= \{<f1_|^C:x,...,fn_J^C:x>| \ x \in D \cap D')\}$ Df. 3.5

    $= <f1_|^C:(D \cap D' \cap d(f1_|^C))),...,fn_J^C:(D \cap D' \cap d(fn_J^C))>$ Df. 3.10

Thus, the equation is correct.

    For the case of $D \subseteq d([f1_|^C,...,fn_J^C]_1$ a simpler equation is used

    $[f1_|^C,...,fn_J^C]_1:D = <f1_|^C:D,...,fn_J^C:D>.$

A proof similar to the above given proof, but based on Definition 3.6.A rather than on Definition 3.10 proves this simpler form of the equation.

Example 3.8. This example is a continuation of Example 3.4. Computations are shown for $r(f_j{}^c)$, for all cases where $d(f_j{}^c) \neq \phi$.

    a) By Definition 3.6.B

    $r([s_1{}^c, tl_1{}^c]_1) = [s_1{}^c, tl_1{}^c]_1 : \, <Z(X.1)>(X)$

        $= \, <s_1{}^c : <Z>, tl_1{}^c : <Z>> \qquad\qquad\qquad$ Lm. 3.5.3

        $= \, <Z(X.1)>, <>> \qquad\qquad\qquad\qquad$ App. E

    b) By similar methods

    $r([s_1{}^c, tl_2{}^c]_1)$

        $= [s_1{}^c, tl_2{}^c]_1 : \, U_i <Z(X.1), Z^i(X.i)>(X)$

        $= \, <s_1{}^c : \, U_i <Z, Z^i>, \, tl_2{}^c : \, U_i <Z, Z^i>>$

        $= \, <Z(X.1), \, U_i <Z^i(X.i+1)>(|X|-1)>$

Lemma 3.5.4. For all f1,f2,f3 in F and for all $f1_i{}^c, f2_j{}^c, f3_k{}^c$, and for all $D \subseteq UD^c$

    1) $(f1_i{}^c \text{-->} f2_j{}^c; f3)_1 : D = f2_j{}^c : (D \cap d(f2_j{}^c) \cap (f1_i{}^c)^{-1} : \{T\})$

    2) $(f1_i{}^c \text{-->} f2; f3_k{}^c)_2 : D = f3_k{}^c : (D \cap d(f3_k{}^c) \cap (f1_i{}^c)^{-1} : \{F\})$

Proof: Proof of case 1 is given. Case 2 is similar.

Let $f_k{}^c = (f1_i{}^c \text{-->} f2_j{}^c; f3)_1$.

By Definition 3.10, $(f1_i{}^c \text{-->} f2_j{}^c; f3)_1 : D$

    $= \{(f1_i{}^c \text{-->} f2_k{}^c; f3)_1 : x | \, x \in D \cap d(f_k{}^c)\}$

    $= \{f2_j{}^c : x | \, x \in D \cap d(f_k{}^c)\} \qquad\qquad$ Df.3.5

    $= f2_j{}^c : (D \cap d(f_k{}^c) \cap d(f2_j{}^c)) \qquad\quad$ Df. 3.10

    $= f2_j{}^c : (D \cap d(f2_j{}^c) \cap (f1_i{}^c)^{-1} : \{T\}) \qquad$ Lm. 3.4.6

Thus, the equation is correct.

    When $D \subseteq d(f_k{}^c)$ then the above equation can be reduced to

$(f1_i{}^c \text{-->} f2_j{}^c; f3)_1 : D = f2_j{}^c : D$.

This simpler equation is used in the example that follows to derive $r(f_j{}^C)$. The more complex form of the equation is used in Theorem 3.5 and later used to extend the characterization of f to include $f_j{}^C$ $\in$ $U_R f_i{}^C$.

Example 3.9. The computations of Example 3.5 are continued, and $r(f_j{}^C)$ is derived from the equation given in the lemma above for $f_j{}^C$: $d(f_j{}^C)$. Only cases where $d(f_j{}^C) \neq \emptyset$ are shown.

    a) By Definition 3.5.B

    $r((atom_1{}^C\text{--->}id_1{}^C;tl)_1)$

        = $(atom_1{}^C\text{--->}id_1{}^C;tl)_1$: Atoms(X)

        = $id_1{}^C$: ATOM(X)   Lm. 3.5.5

        = ATOM(X)   App. E

    b) By similar methods

    $r((atom_2{}^C\text{--->}id;tl_1)_2)$

        = $(atom_2{}^C\text{--->}id;tl_1{}^C)_2$: <Z(X.1)>(X)

        = $tl_1{}^C$: <Z(X.1)>(X)

        = <>

    c) $r((atom_2{}^C\text{--->}id;tl_2{}^C)_2)$

        = $tl_2{}^C$: U_i<Z(X.1),Z^i(X.i+1)>(X)

        = U_i<Z^i(X.i+1)>(|X|-1)

Theorem 3.5 For all f in F, for all $f_j{}^C$, i=1,...,p(f), where $f_j{}^C$ $\in$ $U_R f_i{}^C$, and for all D $\in$ UD^C,

    $f_j{}^C$:D $\in$ UD^C.

Proof: The proof of this theorem is similar to Theorem 3.4. An equation is constructed for $f_j{}^C$:D by recursive applications of the equations given in Lemmas 3.5.2 - 3.5.4 and the set mappings of

primitive functions in Appendix E. For the special case of $D=d(f_j{}^c)$ by

Definition 3.5.3, the equation gives $r(f_j{}^c)$. By Proposition 3.2.4,

$f_j{}^c:U_iD_i = U_if_j{}^c:D_i$ and thus only the case of $D \in UD^c$ needs additional

proof.

case 1) If f is a primitive function in F then by Lemma 3.4.2, Lemma

3.5.1 and Definition 3.10, $f_j{}^c:D \in D^c$.

case 2) If $f=G(f1,...,fn)$ then an equation for $f_j{}^c:D$ can be constructed

by the above described method. The equation has a finite number of

terms and by the method of construction each term is one of the

following:

    i) $f1_j{}^c:(f2_k{}^c:(D \cap d(f2_k{}^c)) \cap d(f1_j{}^c))$

    ii) $<f1_j{}^c:(D\cap d(f1_i{}^c)\cap D'),...,fn_k{}^c:(D\cap d(fn_k{}^c)\cap D')>$

       where $D' = (f1_i{}^c)^{-1}:(O-\underline{i}) \cap ... \cap (fn_k{}^c)^{-1}:(O-\underline{i})$

    iii) $f2_j{}^c:(D \cap d(f2_j{}^c) \cap (f1_i{}^c)^{-1}:\{T\})$

       $f3_k{}^c:(D \cap d(f3_k{}^c) \cap (f1_i{}^c)^{-1}:\{F\})$

By Definition 3.7 and Lemma 3.4.1, $\{T\}$, $\{F\}$, and $O-\underline{i}$ are in UD^c. A

proof by induction is given that $f_j{}^c:D \in UD^c$.

Let S(N) be the statement that "If f is defined by N applications of the

definitions of the functional forms in G, then $f_j{}^c:D \in UD^c$."

Basis: If N=1 then f1,...,fn are primitive functions in F.

case i) By Lemma 3.4.2 and Lemma 3.5.1

    $f1_j{}^c:(f2_k{}^c:(D \cap d(f2_k{}^c)) \cap d(f1_j{}^c)$

reduces to a set in UD^c.

case ii) By Lemma 3.4.2 and Lemma 3.5.1 D', is in UD^c. Then by

Definition 3.7 and Lemma 3.5.1

    $<f1_j{}^c:(D\cap d(f1_j{}^c))\cap D'),...,fn_k{}^c:(D\cap d(fn_k{}^c)\cap D')>$

reduces to a set in UD^c.

case iii) By Lemmas 3.5.1, 3.4.3, and 3.4.2

$$f2_j{}^c:(D \cap d(f2_j{}^c) \cap (f1_i{}^c)^{-1}:\{T\})$$

reduces to a set in UD^c.

Thus, $S(1)$ is true.

Inductive step: Suppose $S(N)$ is true for each $N \geq 1$. If f is defined by at most N applications of the definitions of the functional forms in G, then f1,...,fn are defined by some $M \leq N$ applications of the functional forms in G. By the inductive assumption $f1_j{}^c:D',\ldots,fn_k{}^c:D'$ are in UD^c for all $D' \in UD^c$.

case i) Then by Lemma 3.4.2, $D \cap d(f2_k{}^c)$ is in UD^c and by the inductive assumption

$$f1_j{}^c:(D \cap d(f2_k{}^c))$$

is in UD^c. Then by Lemma 3.4.2

$$f1_j{}^c:(D \cap d(f2_k{}^c) \cap d(f1_j{}^c) \text{ is in } UD .$$

case ii) By Lemma 3.4.2

$$D \cap d(f1_j{}^c) \cap D' \quad \text{and} \quad D \cap d(fn_k{}^c) \cap D'$$

are in UD^c. Then by the inductive assumption and Definition 3.7

$<f1_j{}^c:(D \cap d(f1_j{}^c) \cap D'),\ldots,fn_k{}^c:(D \cap d(fn_k{}^c) \cap D')>$ is in UD^c.

case iii) By Lemma 3.4.2, $D \cap d(f2_j{}^c)$ is in UD^c and by Theorem 3.4 and Lemma 3.4.2

$$D \cap d(f2_j{}^c) \cap (f1_j{}^c)^{-1}:\{T\}$$

is in UD^c. Then by the inductive assumption

$$f2_j{}^c:(D \cap d(f2_j{}^c) \cap (f1_i{}^c)^{-1}: \{T\})$$

is in UD^c. The same proof holds for

$$f3_k{}^c(:D \cap d(f3_k{}^c) \cap (f1_i{}^c)^{-1}: \{F\}).$$

Thus, by induction $f_j{}^c:D$ is in UD^c for all f in F and for all $f_j{}^c$ not in $U_R f_i{}^c$.

For $f_j{}^c$ in $U_R f_i{}^c$, it has been shown that $(f_j{}^c)^{-1}:D$ is not in UD^c.

Thus, Theorem 3.5 completes the characterization of f over O for all f

where any $r \in R$ is not used in the definition of f. Equations have been

given for $d(f_i{}^c)$ and $r(f_i{}^c)$, $i=1,\ldots,p(f)$ by Theorems 3.4 and 3.5

respectively. The observations and methods which follow show that this

characterization can be extended to the general case of f in F.

Consider f in F and $f_j{}^c \in U_R f_i{}^c$.

1) First a set D' is derived such that $d(f_j{}^c) \subseteq D'$, and $D' \in UD^c$.

Construct the equation for $(f_j{}^c)^{-1}:O$ by Theorem 3.4. For each term

$(r_1{}^c)^{-1}:D''$ or $(r_2{}^c)^{-1}:D''$, $r \in R$, that occurs in the equation for $f_j{}^c:D$,

first reduce the term for D''. Then if $D'' \cap r(r_i{}^c)=\phi$, replace the term

for $(r_i{}^c)^{-1}:D''$ by ϕ, and if $D'' \cap r(r_i{}^c) \neq \phi$ replace the term for

$(r_i{}^c)^{-1}:D''$ by the set, <NUM,NUM>. Reduce this modified equation for

$(f_j{}^c)^{-1}:O$. The result is a set D' which must be in UD^c by Theorem 3.4.

2) To derive $r(f_j{}^c)$ construct the equation for $f_j{}^c:D'$, D' derived in

step 1), by Theorem 3.5. Each term of the equation must be one of the

cases i-iii given in Theorem 3.5. Then for any occurrence of a term in

the equation for $f_j{}^c:D$ where the term is in one of the following forms,

   c1) $f2_j{}^c:(D \cap d(f2_j{}^c) \cap (f1_i{}^c)^{-1}:\{T\})$

   c2) $f3_k{}^c:(D \cap d(f3_k{}^c) \cap (f1_i{}^c)^{-1}:\{F\})$

derive $(f1_i{}^c)^{-1}:\{T\}(F)$ by the methods of Theorem 3.4 and Lemma 3.6.1.

Then by the proof of Theorem 3.5, the terms c1 and c2 reduce to a set

in UD^c.

Every occurrence of $r_1{}^c$ or $r_2{}^c$ in the equation for $f_j{}^c:D$, must

occur in a term in the form of c1 or c2 described above or be some

occurrence of

   a) $r_i{}^c:(D'' \cap d(r_i{}^c))$.

where D" is in UD^C. The set D" $\cap$ $d(r_i{}^C)$ is not in UD^C but

$$r_i{}^C:(D \cap d(r_i{}^C)$$

can be reduced to a set in UD^C. By Definition 3.7.B.2 , D" is one of

<{x},NUM>, <NUM,{x}>, <{x},{x'}>, or <NUM,NUM>, for x,x' $\in$ NUM. Thus,

$r_i{}^C:(D" \cap d(r_i{}^C))$ is {T} or {F} or 0 depending on i and D" and can be

reduced without reducing D" $\cap$ $d(r_i{}^C)$, and clearly is in UD^C.

Then, the result is $f_j{}^C:D' = r(f_j{}^C)$ by Theorem 3.5 and Definition 3.10.

3) The derivation of $d(f_j{}^C)$ is similar to the derivation of $r(f_j{}^C)$.

Predicates P_i, i=1,...,n are defined on D', derived in 1), such that:

$$d(f_j{}^C) = \{x \mid x \in D' \ \& \ P_i(x)=True, i=1,...,n\}.$$

as follows.

Consider all primitive functions op $\in$ {eq,$\geq$,>,$\leq$,<,and,or,not}. In the

equation for $f_j{}^C:D'$, replace each occurrence of $op_1{}^C:D"$ or $op_2{}^C:D"$ with

the right hand side of the equation below for the particular case of op.

   a)   for op $\in$ {eq,$\geq$,>,$\leq$,<}

       $op_1{}^C:D" = BOOL(X.1 \ op \ X.2)$

       $op_2{}^C:D" = BOOL(\sim(X.1 \ op \ X.2))$

   b) $not_i{}^C:D" = BOOL(X)$ i=1,2

   c) $and_i{}^C:D" = BOOL(X.1 \ and \ X.2)$ i=1,2   .

   d) $or_i{}^C:D" = BOOL(X.1 \ or \ X.2)$ i=1,2

The right hand side of each of the above equations is of the form

$BOOL(P_i)$ and clearly for either $op_1{}^C$ or $op_2{}^C$, $BOOL(P_i)$ is an equivalent

expression. The cases where (D" $\cap$ <BOOL,BOOL>) = ϕ need no additional

consideration since this implies $f_j{}^C:D'=\phi$.

   Reduce this equation for $f_j{}^C:D'$. For each occurrence of a term, c1

or c2 construct a separate equation for $f1_i{}^C:D"$ by the same method,

replacing each $op_i{}^C$ by the right hand side of the equations for a-d

above. The reduction of each of these equations for $f1_i{}^C:D''$ gives

$BOOL(P_i)$ for some predicate P_i. Define each of these predicates, P_i,

$i=1..n''$, encountered in the reduction of $f_j{}^C:D'$ on D'.

Then the reduction of $f_j{}^C:D'$ gives a set in UD^C with some

predicates, P_i, $i=1,...,n'$, given as the binding of $f_j{}^C:D'$ to D'. For

each of these occurrences of P_i, also define P_i on D'. Each predicate

P_i, $i=1,...,n$, $n=n'+n''$, constructed by either of the above steps is

either True or False for each $x \in D'$ and is True for each x such that x

$\in d(f_j{}^C)$. The resulting expression

   $\{x| \ x \in D' \ \& \ P_i(x)=TRUE, \ i=1,...,n\}$

equals $d(f_j{}^C)$.

Example 3.10.3 which follows shows the computation of $d(f_i{}^C)$ and

$r(f_j{}^C)$ for $f_j{}^C$ in $U_Rd(f_i{}^C)$ by this method. Other examples computed by

the program described in the introduction to this paper are given in the

following chapter. The method gives correct results for all examples

considered.

Example 3.10.

   1) Let $f:x = (\geq \ \text{-->} \ +; \ s1)$. Then the computational forms of f are:

   a) $(\geq_i{}^C \ \text{-->} \ +_1{}^C; \ s1)_1$, $i=1,2$

   b) $(\geq_i{}^C \ \text{-->} \ +_2{}^C; \ 1s)_1$, $i=1,2$

   c) $(\geq_i{}^C \ \text{-->} \ +; \ s_1{}^C)_2$, $i=1,2$

   d) $(\geq_i{}^C \ \text{-->} \ +; \ s_2{}^C)_2$, $i=1,2$

   $f_i{}^C \ \& \ U_Rf_i{}^C$ are also shown

   e) $(\geq_3{}^C \ \text{-->} \ +_i{}^C; \ s_j{}^C)_k$, $i,j,k=1,2$

   f) $(\geq \ \text{-->} \ +; \ 1)_3$.

2) The equations for D', $d(f_j^c) \subseteq D'$, for each f_j^c in a-d are derived from Theorem 3.4.

a) By Lemma 3.3.1.1

$d((\geq_i^c \longrightarrow +_1^c; s)_1)$

$\quad = ((\geq_i^c \longrightarrow +_1^c; s)_1)^{-1}{:}0, \quad i=1,2$

$\quad\quad (\geq_i^c)^{-1}{:}\{T\} \cap (+_1)^{-1}{:}0$ Lm. 3.3.2.4

$D' = \text{<NUM,NUM>} \cap (+_1^c)^{-1}{:}0$ Lm. 3.6.1

$\quad = \text{<NUM,NUM>} \cap \text{<NUM,NUM>}$ App. D

$\quad = \text{<NUM,NUM>}$

b) By the above methods

$d((\geq_i^c \longrightarrow +_2^c; s)_1)$

$\quad = (\geq_i^c)^{-1}{:}\{T\} \cap (+_2^c)^{-1}{:}0, \quad i=1,2$

$D' = \text{<NUM,NUM>} \cap d(+_2^c)$

$\quad = \phi$

c) $d((\geq_i^c \mathbin{--} > +; s_1^c)_2)$

$\quad = (\geq_i^c)^{-1}{:}\{F\} \cap (s_1^c)^{-1}{:}0$

$D' = \text{<NUM,NUM>} \cap (s_1^c)^{-1}{:}0$

$\quad = \text{<NUM,NUM>} \cap U_i\text{<}z^i\text{>}$

$\quad = \text{<NUM,NUM>}$

d) $d((\geq_i^c \longrightarrow +; s_2^c)_2)$

$\quad = (\geq_i^c)^{-1}{:}F \cap (s_2^c)^{-1}{:}0$

$D' = \text{<NUM,NUM>} \cap (\text{Atoms} \cup \underline{\mathbf{l}})$

$\quad = \phi$

By previously demonstrated methods

e) $d((\geq_3^c \longrightarrow +_i^c; s_j^c)_3) = \phi$, for all (i,j,k)

f) $d((\geq \longrightarrow +; s)_3) = d(\geq_3^c)$

3) $f_j{}^C$: D' is shown for cases a-d. The construction of the predicates $P_1,\ldots,P_n$ is shown to the right of the computations of $r(f_j{}^C)$. Only non null cases of D' or $d(f_j{}^C)$ computed in 2) are shown.

a.1) By Lemma 3.5.4.1

$(\geq_1{}^C \!-\!\!-\!\!>\!+_1{}^C; \ s)_1$: $<$NUM,NUM$>$

$= +_1{}^C\!:(<$NUM,NUM$> \ \cap \ d(+_1{}^C) \ \cap \ (\geq_1{}^C)^{-1}\!:\{T\})$

$\quad = +_1{}^C\!: \ (<$NUM(X.1),NUM(X.2)$>$(X) $\cap$ $<$NUM, NUM$>$)

$\quad = +_1{}^C\!: \ <$NUM(X.1),NUM(X.2)$>$(X) $\qquad$ $P_1 = (X.1 \geq X.2)$

$\quad = $ NUM(X.1 + X.2)

a.2) By similar methods

$\quad (\geq_2{}^C \!-\!\!-\!\!>\!+_1{}^C; \ s)_1$: $<$NUM,NUM$>$

$\quad = +_1{}^C\!:(<$NUM(X.1),NUM(X.2)$>$(X) $\cap$ $(\geq_2{}^C)^{-1}\!:\{T\})$

$\quad = +_1{}^C\!:(<$NUM(X.1),NUM(X.2)$>$(X) $\cap$ 0)

$\quad = \phi$

c.1) By Lemma 3.5.4.2

$\quad (\geq_1{}^C \!-\!\!-\!\!>\!+,s_1{}^C)_2$: $<$NUM,NUM$> = \phi$

c.2) By similar methods

$\quad (\geq_2{}^C \!-\!\!-\!\!>\!+;s_1{}^C)_2$: $<$NUM,NUM$>$

$\quad = s_1{}^C\!:(<$NUM,NUM$> \ \cap d(s_1{}^C) \ \cap \ (\geq_2{}^C)^{-1}\!:\{F\})$

$\quad = s_1{}^C\!: \ (<$NUM,(X.1),NUM(X.2)$>$(X) $\cap$ $<$NUM, NUM$>$)

$\quad = s_1{}^C\!:(<$NUM(X.1),NUM(X.2)$>$) $\qquad$ $P_1 = (\sim(X.1 < X.2))$

$\quad = $ NUM(X.1)

4) The domains $d(f_j{}^C)$ are given by the sets D' derived in 2) with the predicates P_i, i=1,...,n, derived in 3), defined on the sets, D', as follows.

a.1) $d((\geq_1{}^C \!-\!\!-\!\!>\!+_1{}^C;s)_1) = <NUM,NUM>$, (X.1 $\geq$ X.2)

c.2) $d((\geq_2{}^C \!-\!\!-\!\!>\!+;s_1{}^C)_2) = <NUM,NUM>$, $\sim$(X.1 $\geq$ X.2)

A set of restrictions of f in F is defined and shown to have the following properties.

1) The set of restrictions is a finite set.

2) The domains of the restrictions in the set partition O, the domain of f.

Equations are given for the domain and range of each restriction in the set.

Since all programs in FP are applications of functions f in F to objects x in O it is reasonable to assume that the characterization of f in F given in this chapter may be useful for reasoning about properties of programs in FP. In the following chapter an execution time cost model is defined for FP, and the methods of this chapter are used to show that the cost of f:x over O can be given as a finite set of costs. In particular it is shown that given x,x' in the domain of some restriction, $cost(f:x) = cost(f:x')$.

CHAPTER IV

EXECUTION TIME COST ANALYSIS FOR FP

Definitions and Preliminaries

In this chapter a method is shown for estimating the execution time
cost of functions f in F over the data domain O. It is similar to
existing methods of cost analysis in several ways.

    1) A computational model is defined which gives the cost of f:x, for

    each f in F and each x in O. The cost of f:x is defined to be the

    number of basic operations performed in the reduction of f:x.

    2) An expression is given for the cost of f:x over the data domain,

    O.

Certain properties of FP require variations from existing methods.
Given the conditional semantics of FP it is not apparent that these
methods can be used to determine all possible computation sequences for
f:x, x in O. However, all functions f in F are total functions,
f:O-->O, and thus a cost must be defined for f:x, for all x in O.
Then a method must be given to derive the possible computation sequences
of f:x and to estimate the cost of f:x for each computation sequence.

The methods of the previous chapter are used to solve these
problems. In particular, it is shown that the possible computation
sequences of f:x are equivalent to the computational restrictions of f.
The following definition defines an execution time cost computational
model for FP.

<u>Definition 4.1.</u> The following execution cost model is defined for FP:

A) The computation sequence for f in F is given by the definition of

f (3). The choice of a particular reduction order for [f1,...,fn]:x

is arbitrary and does not affect the cost of f:x defined below, since

the execution time cost of f:x is defined to be the total number of

operations performed in the reduction of f:x.

B) The symbolic constants $c(f_i)$ denote a symbolic cost associated

with f_i, where f:x $\underline{=}$ $P_i(x)$ --> f_i, i=1,...,flast. The symbolic

constant c(f) denotes a base cost associated with the reduction f:x.

The symbolic constant c(#) denotes the cost of the constant function

in FP.

C) Let f be a function in f defined by

   f:x $\underline{=}$ $P_1(x)$-->f_1;...;$P_{last}(x)$-->f_{last}.

Then for each x in 0 the cost of f:x is denoted by "cost(f:x)" and is

defined to be:

   1) If f is in {atom,null,eq,$\geq$,>,$\leq$,<}, then

      cost(f:x) – c(f)

   Thus, $c(f_i)$–$c(f_j)$, i,j–1,...,flast.

   2) If f is a primitive function not in case 1) then for

   i–1,...,flast,

      $P_i(x)$ & f_i is a constant function ––> cost(f:x)–c(#)

      $P_i(x)$ & f_i is not a constant function ––> cost(f:x)–$c(f_i)$

   It is noted that for all these cases of f except for f–apndl, that

   $c(f_i)$ may unambiguously be denoted by c(f). 3) If f–G(f1,...,fn)

   for some f1,...,fn in F then

      a) cost($\bar{y}$:x) – c(#), where $\bar{y}$:x is the constant function y.

      b) cost((f1$\oplus$f2):x) – c($\oplus$) + cost(f2:x) + cost(f1:(f2:x))

c) cost([f1,...,fn]:x) = c([]) + cost(f1:x) +...+ cost(fn:x)

d) $P_1(x)$ ==> cost((f1-->f2;f3):x) = c(-->) + cost(f1:x)

+ cost(f2:x)

$P_2(x)$ ==> cost((f1-->f2;f3):x) = c(-->) + cost(f1:x)

+ cost(f3:x)

$P_3(x)$ ==> cost((f1-->f2;f3):x) = c(-->) + cost(f1:x)

The following observations are made about the execution cost model

of Definition 4.1.

1) For every f in F if $P_i(x)$=True and f_i is not a constant function

then the actual execution time of f:x for any particular machine

implementation may vary significantly over x in 0. This level of

inaccuracy in estimating the cost(f:x) cannot be avoided unless

implementation dependent assumptions are introduced into the model.

2) No cost is assigned to the operations required to determine the

case of $P_i(x)$. The order given for f_i, i=1,...,flast in Appendix A

and by Backus (3) is clearly not optimal for minimizing the number of

these operations required. Including them in cost(f:x) would require

assumptions about the average case of x for f:x, or would require

that cost be defined as worst or best case cost. For these reasons

and for the sake of simplicity in the examples of this chapter, they

are not included in Definition 4.1.

3) For the case of f in 1) above cost(f:x) is given as c(f) even

though f_i, i=1,...,flast is a constant function. This choice is made

arbitrarily to reflect patterns of cost in cost(f:x). For all cases

of c(f) or c(#), $c(f_i)$ could be substituted without affecting the

methods described later in this chapter.

The model of Definition 4.1 is independent of any particular machine implementation, except for the cases described in 3) above. The choices for the cost functions given are made arbitrarily for the sake of simplicity or to maintain machine independence in the model. Other cost functions could be defined to reflect some particular hardware or software implementation. For a developing language such as FP, both machine independence and flexibility are desirable properties of an execution time cost model.

Methods for Estimating Execution Time Cost

In this section methods are given for estimating the cost of $f:x$ over the data domain, O. In particular, the results of the previous chapter are used to show that a finite set of cost functions under Definition 4.1 gives the cost of $f:x$ over O.

<u>Theorem 4.1.</u> For all f in F and for all $f_i{}^C$, $i=1,\ldots,p(f)$,

   If x',x'' are in $d(f_i{}^C)$

   then $cost(f:x') = cost(f:x'')$.

<u>Proof:</u> A proof of this theorem is given in Appendix F.

<u>Definition 4.2.</u> The notation of Definition 4.1 is extended as follows. Let

   $c(f_i{}^C)$ denote $cost(f:x)$ for all x in $d(f_i{}^C)$.

This extension implies no contradictions of Definition 4.1 since $c(f)$ and $c(d(f_i{}^C))$ are both cost functions under the definition.

Methods are shown in Chapter 3 to derive $d(f_i{}^C)$ and $r(f_i{}^C)$ for all f in F and for each $f_i{}^C$, $i=1,\ldots,p(f)$. By Theorem 3.2 $p(f)$ is finite and by Theorem 4.1 $c(f_i{}^C)$ is a single cost function under Definition

4.1. Therefore, cost(f:x) for all x in O is given by $c(f_i^c)$, i=1,...,p(f).

Theorem 4.1 also shows that $c(f_i^c)$ can be computed directly from f_i^c. In the examples that follow, $r(f_i^c)$ and $c(f_i^c)$ are computed in parallel to illustrate this for one case of each f in G.

Example 4.1. Let f:x=(setl):x and Z=O-$\underline{i}$. Computations are shown for $f_i^c:d(f_i^c)$, and for $c(f_i^c)$, f_i^c, i=1,...,p(f) except where $d(f_i^c) = \phi$.

$$r(f_i^c) = f_i^c: \quad d(f_i^c) \qquad\qquad c(f_i^c)$$

1) $r((s_1{}^c \oplus tl_2{}^c)_1{}^c)$

$= s_1{}^c:(tl_2{}^c: U_i < Z(X.1),Z^i(X.i+1)>(X)$

$\qquad\qquad\qquad\qquad\qquad c(\oplus) + cost(tl_2) + cost(s_1)$

$= s_1{}^c:(U_i < Z^i(X.i+1)>(|X|-1) \quad c(\oplus) + c(tl) + cost(s_1)$

$= Z(X.2) \qquad\qquad\qquad c(\oplus) + c(tl) + c(s)$

2) $r((s_2{}^c \oplus tl_1{}^c)_1{}^c)$

$= s_2:(tl_1{}^c: < Z(X.1)>(X) \qquad c(\oplus) + cost(tl_1) + cost(s_2)$

$= s_2:(<>) \qquad\qquad\qquad c(\oplus) + c(*) + cost(s_2)$

$= \underline{i} \qquad\qquad\qquad\qquad c(\oplus) + c(*) + c(\#)$

3) $r((s_2{}^c \oplus tl_3{}^c)_1{}^c)$

$= s_2{}^c:(tl_3{}^c:(Atoms \cup \underline{i})) \quad c(\oplus) + cost(tl_3) + cost(s_2)$

$= s_2{}^c:(\underline{i}) \qquad\qquad\qquad c(\oplus) + c(*) + cost(s_2)$

$= \underline{i} \qquad\qquad\qquad\qquad c(\oplus) + c(*) + c(*)$

Example 4.2. Let $f: x = [s, tl] : x$ and $Z = 0 - \underline{|}$.

$$r(f_i^C) = f_i^C: \quad d(f_i^C) \qquad\qquad c(f_i^C)$$

1) $r([s_1^C, tl_1^C]_1)$

   $= \langle s_1^C: \langle Z(X.1)\rangle(X), tl_1^C: \langle Z(X.1)\rangle(X)\rangle$

   $\qquad\qquad\qquad c([]) + cost(s_1) + cost(tl_1)$

   $= \langle Z(X.1), \langle\rangle\rangle \qquad c([]) + c(s) + c(\#)$

2) $r([s_1^C, tl_2^C]_1)$

   $= \langle s_1^C: U_i \langle Z(X.1), Z^i(X.i+1)\rangle(X), tl_2^C: U_i \langle Z(X.1), Z^i(X.i+1)\rangle(X)\rangle$

   $\qquad\qquad\qquad c([]) + cost(s_1) + cost(tl_2)$

   $= \langle Z(X.1), U_i \langle Z^i(X.i+1)\rangle(\underline{|}X\underline{|}-1)\rangle$

   $\qquad\qquad\qquad c([]) + c(s) + c(tl)$

3) $r([s_2^C, tl_3^C]_2)$

   $= \langle s_2^C: (Atoms\ U\ \underline{|}), tl_3^C(Atoms\ U\ \underline{|})\rangle$

   $\qquad\qquad\qquad c([]) + cost(s_2) + cost(tl_3)$

   $= \langle\underline{|}, \underline{|}\rangle \qquad c([]) + c(\#) + c(\#)$

   $= \underline{|}$

Example 4.3. Let $f: x = (atom \longrightarrow id; tl): x$ and $Z = 0 - \underline{|}$.

$$r(f_i^C) = f_i^C: \quad d(f_i^C) \qquad\qquad c(f_i^C)$$

1) $r((atom_1^C \longrightarrow id_1^C; tl)_1)$

   $= ((atom_1^C: Atoms) \longrightarrow id_1^C; tl): Atoms(X)$

   $\qquad\qquad\qquad c(\longrightarrow) + cost(atom_1) + cost(id_1)$

   $= (T \longrightarrow id_1; tl)_1^C: Atoms \quad c(\longrightarrow) + c(atom) + cost(id_1)$

   $= id_1^C: Atoms(X)$

   $= Atoms(X) \qquad\qquad c(\longrightarrow) + c(atoms) + c(id)$

2) $r((atom_2{}^C \text{-->} id; tl_1{}^C)_2)$

$\quad = ((atom_2{}^C : <Z(X.1)>(X)) \text{-->} id;\ tl_1{}^C)_2{}^C :\ <Z(X.1)>(X)$

$$c(\text{-->}) + cost(atom_1) + cost(tl_1)$$

$\quad = (F \text{-->} id; tl_1{}^C)_2{}^C :\ <Z(X.1)>(X)$

$$c(\text{-->}) + c(atom) + cost(tl_1)$$

$\quad = tl_1{}^C :\ <Z(X.1)>(X)$

$\quad = <>$ $\qquad\qquad\qquad\quad c(\text{-->}) + c(atom) + c(\#)$

3) $r((atom_2{}^C \text{-->} id; tl_2{}^C)_2)$

$\quad = ((atom_2{}^C : U_i<Z,Z^i>) \text{-->} id; tl) :_2{}^C :\ U_i<Z,Z^i>$

$$c(\text{-->}) + cost(atom_2) + cost(tl_2)$$

$\quad = (F \text{-->} id; tl_2{}^C)_2{}^C :\ U_i<Z(X.1),Z^i(X.i+1)>(X)$

$$c(\text{-->}) + c(atom) + cost(tl_2)$$

$\quad = tl_2{}^C :\ U_i<Z(X.1),Z^i(X.i+1)>(X)$

$\quad = U_i<Z^i(X.i+1)>(|X|-1)$ $\qquad c(\text{-->}) + c(atom) + c(tl)$

4) $r((atom_3{}^C \text{-->} id; tl)_3)$

$\quad = ((atom_3{}^C : \underline{\bot}) \text{-->} id; tl)_3{}^C :\ \underline{\bot}$

$$c(\text{-->}) + cost(atom_3)$$

$\quad = (\underline{\bot} \text{-->} id; tl)_3{}^C :\ \underline{\bot}$

$$c(\text{-->}) + c(atom)$$

$\quad = \underline{\bot}$

The examples above show the computation of cost(f:x) over O for simple cases of f in F. The tables of Appendix G show $d(f_i{}^C)$, $r(f_i{}^C)$, and $c(f_i{}^C)$ for more complex cases of f. These values were computed by the program described in the introduction of this paper, except for the cases of $r(f_i{}^C) = \underline{\bot}$ which were computed by hand. For each f, $U_i d(f_i{}^C)$ such that $r(f_i{}^C) = \underline{\bot}$ is given as a single set, and a single cost function,

cost(f:x), is given which is the max(cost(f:x)) for all x such that
f:x=$\underline{\iota}$.

A machine independent model for FP is defined which gives the cost
of f:x for all f in F and for each x in O. The methods of Chapter 3 are
then used to show that the cost of f:x over O can be given as a finite
set of cost functions derived from this definition. Computations of the
cost of f:x over O are shown for simple cases of f in F, and computed
results are given for more complex cases of f.

CHAPTER V

SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

Summary

A set of functions, F, is defined which is a subset of functions in
FP as defined by Backus (3). The set contains functional forms for
construction, conditional, and composition and contains a subset of
the primitive functions defined by Backus (3) and Williams (11). Then
for each function f in F a set of computational restrictions of f is
defined. Definition 3.4 gives a method to derive the computational
restrictions of f in the general case and Theorem 3.4 proves that this
set of restrictions is finite. The examples of Chapter 3 and the
computed results of the program described in the introduction of this
thesis verify that the set of restrictions can be computed.

In Theorem 3.3 it is proved that the domains of the computational
restrictions of f partition O, the domain of f. Thus, every data object
in O is in the domain of one and only one computational restriction.
Theorems 3.4 and 3.5 show that an equation can be constructed for the
domain and the range of each computational restriction of f. The
methods for constructing the equations are proven to be correct for a
significant subset of F, and for the remaining cases examples are shown.
The computed results in Appendix G give further verification of the
methods and show that the methods can be automated. Thus, a method has
been shown for computing a nontrivial characterization of functions in F.

In Chapter 4 this characterization of functions for FP is shown to be useful. An execution time cost model is defined which gives an estimate of the execution time cost of f:x for all f in F and for each x in O. The model is derived directly from the definition of FP (3). It is shown to be a machine independent model and also to have a flexible framework which may be used to define an execution time cost model for any particular hardware or software implementation of FP.

Theorem 4.1 shows that the set of computational restrictions of each function f in F correspond to the possible computation sequences of f. A proof is given that a single cost function derived under the model gives the cost of f:x for all x in the domain of some computational restriction. Then since the set of computational restrictions is finite, the cost of f over O is given by a finite set of cost functions. In Chapter 3 examples are shown of the computation of the execution time cost of f over O for simple cases of f. Then for more complex cases of f, the domain, range and cost of each restriction of f are given in table form in Appendix G. These results were computed by the program described in the introduction to this thesis. All computed results are consistent with the expected results given by the theory and methods.

Conclusions

FP is a functional programming language in which all programs are the application of a function to an object. Existing methods for characterizing programs and existing methods of cost analysis for conventional languages do not give any apparent solution to the problem of execution time cost analysis for functional languages. A machine

independent execution cost model based on existing methods of cost analysis is defined for FP in Definition 4.1. Then the characterization of functions given in this thesis is combined with this cost model to give a method for execution time cost analysis for FP. The examples given in Chapter 4 and the computed results given in Appendix G verify this method of cost analysis for a significant subset of FP.

In particular, this characterization of functions gives the finite set of possible computation sequences for each function and the domain and range of each computation sequence. A formal specification is given by Definition 3.7 for the domains and ranges of these computation sequences. The examples of Chapters 3 and 4 and the computed results in Appendix G show that these sets can be computed for the general case of f in F. Then by comparing the respective domains and ranges of any two functions in F it is possible to determine if the two functions are equivalent. The computed results of Appendix G show the domains and ranges for several cases of functions which are equivalent (3). In all of these cases the computed domains and ranges of equivalent functions are equivalent. Thus, it is reasonable to assume that the characterization of functions given in this thesis might be useful for examining other properties of programs in FP such as program equivalence.

Recommendations

The characterization of functions given in this paper is proven to be correct for a significant subset of F, the set of functions in FP. The examples shown indicate that the equations constructed for the domains and the ranges of the computation sequences of the functions are

correct for the general case, but a formal proof is needed. The set F as defined for this paper does not include programs which contain iteration or recursion. Given the consistency of the results obtained for the subset of FP considered and assuming the proof described above is given, then it is clear that extending the methods of this paper to a larger class of functions is a promising course for future work.

Execution time cost analysis is a desirable tool in the developmental stage of a programming language as well as a necessary tool in a production level language. One valuable use for automated techniques of cost analysis is program optimization. The proofs of Chapter 3 and the computed results of Appendix G show that the method can be used to determine the equivalence of two functions in F. Combined with the cost analysis model this gives the potential for automated optimization of programs. Thus, another promising course for future work is the application of the methods to some particular hardware and software implementation of FP.

The characterization of functions given in this paper clearly has other applications distinct from cost analysis of functional languages. For procedural languages techniques exist to formally specify the relationship between the input and the output of programs. These methods are used to give proof of correctness of programs, but they are limited in that heuristic input about program intent is required (12). The methods of this paper clearly meet the requirements for formally specifying the results of a program over all possible inputs without the above described limitations.

The possible applications of the methods described in this paper are wide ranging. For the subset of FP considered they give a useful

tool for future work in the development of functional languages. If the methods can be applied to a larger class of functions, then some of the limitations of conventional algebraic languages may be shown to not apply to functional languages.

BIBLIOGRAPHY

(1) Aho, A. V., J. E. Hopcroft, and J. D. Ullman. The Design and
        Analysis of Computer Algorithms. Reading: Addison-Wesley
        Publishing Co., 1974.

(2) Aho, A. V., R. Sethi, and J. D. Ullman. Compilers: Principles,
        Techniques, and Tools. Reading: Addison-Wesley Publishing
        Co., 1986.

(3) Backus, J. "Can Programming Be Liberated From the Von Neumman
        Style?" CACM, 21, 8(August), pp 1-10.

(4) Backus, J. "Function Level Programs as Mathematical Objects."
        Proceedings of the ACM Conference on Functional Programming
        Languages and Computer Architecture. Portsmouth, N.H,
        October, 1981, pp 1-10.

(5) COSERS: The Computer Science and Engineering Research Study.
        edited by Bruce W. Arden, Cambridge: The MIT Press, 1980, pp
        137-295.

(6) Hoare, C. "An Axiomatic Basis for Computer Programming." CACM,
        12, 10(October), pp 576-583.

(7) Katayama, T. "Type Inference and Type Checking for Functional
        Programming Languages: a Reduced Computation Approach."
        Conference Record of the 1984 Symposium on Lisp and Functional
        Programming. 1984, pp 263-272.

(8) Manna, Z., S. Ness, and J. Vuillemin. "Inductive Methods for
        Proving Properties of Programs." CACM, 16, 8(August), pp
        491-502.

(9) McCarthy, J. "Recursive Functions of Symbolic Expressions and
        Their Computation by Machine, Part I." CACM, 3, 4(April), pp
        184-195.

(10) Simmons, George F. Introduction to Topology and Modern
        Analysis. New York: McGraw Hill, 1963.

(11) Williams J. "Notes on the FP Style of Functional
        Programming." Proceedings of the 1981 Conference on
        Functional Languages. 1981, pp 73-101.

(12) Wulf, W. A., M. Shaw, P. N. Hilflinger, and L. Fion.
        Fundamental Structures of Computer Science. Reading:
        Addison-Wesley Publishing Co., 1981.

DEFINITIONS OF FUNCTIONS IN F

Primitive Functions

For all x in O

1) $s:x \equiv (x=<x_1,\ldots,x_n>, n\geq1) \longrightarrow x_1; \underline{\bot}$

2) $id:x \equiv x$

3) $tl:x \equiv (x=<x_1>) \longrightarrow <>; (x=<x_1,\ldots,x_n>, n\geq2) \longrightarrow <x_2,\ldots,x_n>; \underline{\bot}$

4) $atom:x \equiv (x$ is an atom$) \longrightarrow T; (x=<x_1,\ldots,x_n>, n\geq1) \longrightarrow F; \underline{\bot}$

5) $null:x \equiv (x=<>) \longrightarrow T; \sim(x=<> ! x=\underline{\bot}) \longrightarrow F; \underline{\bot}$

6) $rev:X \equiv (x=<>) \longrightarrow <>; (x=x_1,\ldots,x_n>, n\geq1) \longrightarrow (<x_n,\ldots,x_1>; \underline{\bot}$

7) $apndl:x \equiv (x=<x_1,<>>) \longrightarrow <x_1>;$

$(x=<x_1,<x_2,\ldots,x_n>>) \longrightarrow <x_1,\ldots,x_n>; \underline{\bot}$

8) $and:x \equiv (x=<T,T>) \longrightarrow T;$

$(x=<T,F> ! x=<F,T> ! x=<F,F>) \longrightarrow F; \underline{\bot}$

9) $or:x \equiv (x=<T,T> ! x=<T,F> ! x=<F,T>) \longrightarrow T;$

$(x=<F,F>) \longrightarrow F; \underline{\bot}$

10) $not:x \equiv (x=T) \longrightarrow F; (x=F) \longrightarrow T; \underline{\bot}$

11) For op in $\{+,*,sub,div\}$

$op:x \equiv (x=<x_1,x_2>, x_1,x_2$ are numbers$) \longrightarrow (x_1$ op $x_2); \underline{\bot}$

12) For r in $R= \{eq,\geq,>,\leq,<\}$

$r:x \equiv (x=<x_1,x_2>, x_1,x_2$ are numbers, $x_1 r x_2) \longrightarrow T;$

$(x=<x_1,x_2>, x_1,x_2$ are numbers, $x_1 \sim r x_2) \longrightarrow F; \underline{\bot}$

Functional Forms

For all x,y in O and all f1,...,fn in F

    1) (f1$\circ$f2):x $\equiv$ f1:(f2:x)

    2) [f1,...,fn]:x $\equiv$ ~(f1:x=$\underline{|}$!...! fn:x=$\underline{|}$) --> <f1:x,...,fn:x>; $\underline{|}$

    3) (f1-->f2;f3):x $\equiv$ (f1:x=T) --> f2:x; (f1:x=F) --> f3:x; $\underline{|}$

    4) $\overline{x}$:y $\equiv$ ~(y=$\underline{|}$) --> x; $\underline{|}$

APPENDIX B

COMPUTATIONS OF f:x

The examples below are computations of f:x for various cases of x
in O and f in F.

1) f:x = (≥-->+,s):x

   a) (≥-->+;s): <5,4>

     = ((≥:<5,4>)-->+;s): <5,4>

     = (T-->+;s): <5,4>

     = +: <5,4>

     = 9

   b) (≥-->+;s): <5,4,3>

     = ((≥:<5,4,3>)-->+;s): <5,4,3>

     = (⊥-->+;s): <5,4,3>

     = ⊥

2) f:x = (eq⊕apndl):x

   a) (eq⊕apndl): <6,<2>>

     = eq:(apndl:<6,<2>>)

     = eq:(<6,2>)

     = F

   b) (eq⊕apndl): <6,2>

     = eq:(apndl:<6,2>)

     = eq:(⊥)

     = ⊥

3) f:x = [+,tl]:x

   a) [+,tl]: <2,3>

      = <+:<2,3>, tl:<2,3>>

      = <5,<3>>

   b) [+,tl]: <2>

      = <+:<2>, tl:<2>>

      = <$\underline{\perp}$, <>>

      = $\underline{\perp}$

4) f:x = ((≥⊕[s,10])--> sub⊕[s,1]; tl):x

   a) ((≥⊕[s,10])--> sub⊕[s,1]; tl): <13, AB>

      = ((≥⊕[s,10]:<13,AB>)--> sub⊕[s,1]; tl): <13,AB>

      = (≥:([s,10]:<13,AB>)--> sub⊕[s,1]; tl): <13,AB>

      = (≥:(<s:<13,AB>,10:<13,AB>)-->sub⊕[s,1];tl):<13,AB>

      = (≥:(<13,10>)--> sub⊕[s,1]; tl): <13,AB>

      = T --> sub⊕[s,1]; tl): <13,AB>

      = sub⊕[s,1]: <13,AB>

      = sub:([s,1]:<13,AB>)

      = sub:(<s:<13,AB>, 1:<13,AB>>)

      = sub:(<13,1>)

      = 12

   b) ((≥⊕s,10])--> sub⊕[s,1]; tl): <AB,13>

      = ((≥⊕[s,10]:<AB,13>)--> sub⊕[s,1]; tl): <AB,13>

      = (≥:([s,10]:<AB,13>)--> sub⊕[s,1]; tl): <AB,13>

      = (≥:(<s:<AB,13>,10:<AB,13>)-->sub[s,1];tl):<AB,13>

      = ((≥:<AB,10>)--> sub[s,1]; tl): <AB,13>

      = $\underline{\perp}$

DOMAINS AND RANGES OF COMPUTATIONAL RESTRICTIONS

OF PRIMITIVE FUNCTIONS IN F

The domain and range of each canonical restriction are given for each primitive function in F. The sets are given in the closed form of Definition 3.7. Let $Z = O - \bot$.

f_i^c	$d(f_i^c)$	$r(f_i^c)$
s_1^c	$U_i <Z^i>$	Z
s_2^c	Atoms U $\bot$	$\bot$
id_1^c	O	O
tl_1^c	$<Z>$	$<>$
tl_2^c	$U_i <Z,Z^i>$	$U_i <Z^i>$
tl_3^c	Atoms U $\bot$	$\bot$
$atom_1^c$	Atoms	T
$atom_2^c$	$U_i <Z^i>$	F
$atom_3^c$	$\bot$	$\bot$
$null_1$	$<>$	T
$null_2^c$	$O - \{<>, \bot\}$	F
$null_3^c$	$\bot$	$\bot$

rv_1^c <>   <>

rv_2 $U_I <Z^I>$ $U_I <Z^I>$

rv_3^c (Atoms-<>) U $\underline{\bot}$ $\underline{\bot}$

$apndl_1^c$ <Z,<>>   <Z>

$apndl_2^c$ $<Z,U_I <Z^I>>$ $U_I <Z,Z^I>$

$apndl_3^c$ Atoms U $\underline{\bot}$ U<Z>

       U $<Z,Atoms-<>> U_I <Z,Z,Z^I>$ $\underline{\bot}$

and_1^c <T,T>   T

and_2^c <T,F> U <F,T> U <F,F>   F

and_3^c Atoms U $\underline{\bot}$ U <Z> U <Z-BOOL,Z>

       U <Z,Z-BOOL> U <Z-BOOL,Z-BOOL>

       $U_I <Z,Z,Z^I>$ $\underline{\bot}$

or_j^c and not_j^c, j=1..3 are similar to and

for op G {*, +, sub, div}

op_1^c <NUM,NUM>   NUM

op_2^c Atoms U $\underline{\bot}$ U <Z> U <Z,Z-NUM>

       U <Z-NUM,Z> U <Z-NUM,Z-NUM>

       $U_I <Z,Z,Z^I>$ $\underline{\bot}$

for r G R={eq,$\geq$,>,$\leq$,<}

r_1^c (r, <NUM,NUM>)   T

r_1^c ($\sim$r, <NUM,NUM>)   F

r_3^c Atoms U $\underline{\bot}$ U <Z> U <Z,Z-NUM>

       <Z-NUM,Z-NUM> U <Z-NUM,Z-NUM>

       $U_I <Z,Z,Z^I>$ $\underline{\bot}$

APPENDIX D

INVERSE SET MAPPING EQUATIONS FOR

THE COMPUTATIONAL RESTRICTIONS OF

PRIMITIVE FUNCTIONS IN F

For all canonical restrictions $f_j{}^c$, $j=1..flast$ an equation is given

for $(f_j{}^c)^c-1:D$, where D is any subset of $r(f_j{}^c)$. For the cases where

$r(f_j{}^c)$ is a constant, $(f_j{}^c)^{-1}:D = d(f_j{}^c)$, and the set $d(f_j{}^c)$ is given in

Appendix C. In these cases $r(f_j{}^c)$ is a singleton set. Let $Z=0-\underline{\mathrm{I}}$ and D

$\in D^c$.

$$(s_1{}^c)^{-1}:D = D \subseteq Z \longrightarrow U_I<D,Z^I>$$

$$(s_2{}^c)^{-1}:D = D = \{\underline{\mathrm{I}}\} \longrightarrow d(s_2{}^c)$$

$$(id_1{}^c)^{-1}:D = D \subseteq 0 \longrightarrow D$$

$$(tI_1{}^c)^{-1}:D = D = \{<>\} \longrightarrow d(tI_1{}^c)$$

$$(tI_2{}^c)^{-1}:D = D \subseteq U_I<Z^I> \longrightarrow U_I<Z,D_1,\ldots,D_k{}^I,\ldots,D_n>$$

$$(tI_3{}^c)^{-1}:D = D = \{\underline{\mathrm{I}}\} \longrightarrow d(tI_3{}^c)$$

$$(atom_1{}^c)^{-1}:D = D = \{T\} \longrightarrow d(atom_1{}^c)$$

$$(atom_2{}^c)^{-1}:D = D = \{F\} \longrightarrow d(atom_2{}^c)$$

$$(atom_3{}^c)^{-1}:D = D = \{\underline{\mathrm{I}}\} \longrightarrow d(atom_3{}^c)$$

$$(null_1{}^c)^{-1}:D = D = \{T\} \longrightarrow d(null_1{}^c)$$

$$(null_2{}^c)^{-1}:D = D = \{F\} \longrightarrow d(null_2{}^c)$$

$$(null_3{}^c)^{-1}:D = D = \{\underline{\mathrm{I}}\} \longrightarrow d(null_3{}^c)$$

$(rev_1{}^c)^{-1}:D \equiv D = \{<>\} \longrightarrow d(rev_1{}^c)$

$(rev_2{}^c)^{-1}:D \equiv D \subseteq U_|<Z^|> \longrightarrow U_|<D_n,\ldots,D_k{}^|,\ldots,D_1>$

$(rev_3{}^c)^{-1}:D = D \subseteq (Atoms-<>U-|) \longrightarrow d(rev_3{}^c)$

$(apndl_1{}^c)^{-1}:D \equiv D \subseteq <Z> \longrightarrow <D_1,<>>$

$(apndl_2{}^c)^{-1}:D \equiv D \subseteq U_|<Z,Z^|> \longrightarrow <D_1,U_|<D_2,\ldots,D_k{}^|,\ldots,D_n>>;\underline{|}$

$(apndl_3{}^c)^{-1}:D \equiv D = \{\underline{|}\} \longrightarrow d(apndl_3{}^c)$

$(and_1{}^c)^{-1}:D \equiv D = \{T\} \longrightarrow d(and_1{}^c)$

$(and_2{}^c)^{-1}:D \equiv D = \{F\} \longrightarrow d(and_2{}^c)$

$(and_3{}^c)^{-1}:D \equiv D = \{\underline{|}\} \longrightarrow d(and_3{}^c)$

$(or_j{}^c)^{-1}:D$ and $(not_j{}^c)^{-1}:D$, $j=1,\ldots,3$ are similar.

For op G {*, +, sub, div},

$(op_1{}^c)^{-1}:D \equiv D \subseteq NUM \longrightarrow <NUM,NUM>$

$(op_2{}^c)^{-1}:D \equiv D = \{\underline{|}\} \longrightarrow d(op_2{}^c)$

For r G R, $(r_i{}^c)^{-1}:D$, $i=1,2$ is not in D^c and is

is give as a relation on <NUM,NUM>

$(r_1{}^c)^{-1}:D \equiv D = \{T\} \longrightarrow (r, <NUM,NUM>)$

$(r_2{}^c)^{-1}:D \equiv D = \{F\} \longrightarrow (\tilde{r}, <NUM,NUM>$

$(r(_3{}^c)-1:D \equiv D = \{\underline{|}\} \longrightarrow d(r_3{}^c)$

SET MAPPING EQUATIONS FOR THE

COMPUTATIONAL RESTRICTIONS OF

PRIMITIVE FUNCTIONS IN F

For all $f_j{}^c$, $j=1,...r$, flast an equation is given for $f_j{}^c:D$, where D _ $d(f_j{}^c)$. The equations are defined for all $D \in D^c$ except where noted. Let $Z = O-\underline{\iota}$, and $D \in D^c$.

$s_1{}^c:D \doteq D \subseteq U_i<Z^i> \longrightarrow D_1(X.1)$

$s_2{}^c:D \doteq D \subseteq (Atoms \cup \underline{\iota}) \longrightarrow \{\underline{\iota}\}$

$id_1{}^c:D \doteq D \subseteq O \longrightarrow D$

$tl_1{}^c:D \doteq D \subseteq <Z> \longrightarrow \{<>\}$

$tl_2{}^c:D \doteq D \subseteq U_i<Z,Z^i>$

$\longrightarrow U_i<D_2(X.2),...,D_k{}^i(X.k+i-1),...,D_n(X.last)>(|X|-1)$

$tl_3{}^c:D \doteq D \subseteq Atoms \cup \underline{\iota} \longrightarrow \{\underline{\iota}\}$

$atom_1{}^c:D \doteq D \subseteq Atoms \longrightarrow \{T\}$

$atom_2{}^c:D \doteq D \subseteq U_i<Z^i> \longrightarrow \{F\}$

$atom_3{}^c:D \doteq D = \{\underline{\iota}\} \longrightarrow \{\underline{\iota}\}$

$null_1{}^c:D \doteq D = \{<>\} \longrightarrow \{T\}$

$null_2{}^c:D \doteq D \subseteq (O - \{<>,\underline{\iota}\}) \longrightarrow \{F\}$

$null_3{}^c:D \doteq D = \{\underline{\iota}\} \longrightarrow \{\underline{\iota}\}$

$rev_1{}^c:D \equiv D = \{<>\} \longrightarrow \{<>\}$

$rev_2{}^c:D \equiv D \subseteq U_i<Z^i>$

$\qquad \longrightarrow U_i<D_n{}^i(X.last),\ldots,D_k{}^i(X.last-(n-k)-i+1),\ldots,D_1(X.1)>(X)$

$rev_3{}^c:D \equiv D \subseteq ((Atoms - <>) \cup \underline{i}) \longrightarrow \{\underline{i}\}$

$apndl_1{}^c:D \equiv D \subseteq <Z,\{<>\}> \longrightarrow <D_1(X.1)>$

$apndl_2{}^c:D \equiv D \subseteq <Z,U_i<Z^i>> \longrightarrow U_i<D_1(X.1),D_{21}(X.2.1),\ldots$

$\qquad ,D_{2k}{}^i(X.2.k+i-1),\ldots,D_{2n}(X.2.n)>(|X.2|+1)$

$apndl_3{}^c:D \equiv D \subseteq (Atoms \cup \underline{i} \cup<Z> \cup <Z,Atoms-\underline{i}> \cup_i<Z,Z,Z^i>) \longrightarrow \{\underline{i}\}$

$and_1{}^c:D \equiv D = <\{T\},\{T\}> \longrightarrow \{T\}$

$and_2{}^c:D \equiv D \subseteq \{<\{T\},\{F\}> \cup <\{F\},\{T\}> \cup <\{F\},\{F\}>\} \longrightarrow \{F\}$

$and_3{}^c:D \equiv D \subseteq (Atoms \cup \underline{i} \cup <Z> \cup <Z,Z-BOOL> \cup <Z-BOOL,Z>$

$\qquad \cup <Z-BOOL,Z-BOOL> \cup_i<Z,Z,Z^i>) \longrightarrow \{\underline{i}\}$

$or_j{}^c:D$ and $not_j{}^c:D$, $j=1,\ldots,3$ are similar

For op $\in \{*,+,sub,div\}$

$op_1{}^c:D \equiv D \subseteq <NUM,NUM> \longrightarrow NUM((X.1) + (X.2))$

$op_2{}^c:D \equiv D \subseteq (Atoms \cup \{\underline{i}\} \cup <Z> \cup <Z,Z-NUM> \cup <Z-NUM,Z>$

$\qquad \cup <Z-NUM,Z-NUM> \cup_i<Z,Z,Z^i>) \longrightarrow \{\underline{i}\}$

For $r \in R$

$r_1{}^c:D \equiv D \subseteq (r,<NUM,NUM>) \longrightarrow \{T\}$

$r_2{}^c:D \equiv D \subseteq (\tilde{}r,<NUM,NUM>) \longrightarrow \{F\}$

$r_3{}^c:D \equiv D \subseteq (Atoms \cup \underline{i} \cup<Z>, <Z,Z-NUM> \cup <Z-NUM,Z>$

$\qquad \cup <Z-NUM,Z-NUM> \cup_i<Z,Z,Z^i>) \longrightarrow \{\underline{i}\}$

APPENDIX F

THEOREM 4.1

Theorem 4.1. For all f in F and for all $f_i{}^C$, i=1,...,p(f),

   if x',x" $\in$ d($f_i{}^C$)

   then cost(f:x')=cost(f:x")

where cost(f:x) is defined in Definition 4.1.

Proof: Let f be in F, and x',x" be in d($f_i{}^C$). The various cases of f

are proved below. The notation for $P_i(x)$ is extended to show the

particular case of f as follows. For f_i:x $\equiv$ $P_i(x)$-->f_i, i=1..flast,

denote $P_i(x)$ by

   $P_i(f,x)$.

case 1) If f is a primitive function in F, then the canonical

restrictions of f are

   $f_i{}^C$:x = f_i:x = $P_i(x)$ --> f_i, i=1,...,flast.

Since x',x" $\in$ d($f_i{}^C$) for one and only one i, then $P_i(f,x')$=True and

$P_i(f,x")$=True for one and only one i. Then by Definition 4.1 either

   a) cost(f:x') = cost(f:x") = c(#), or

   b) cost(f:x') = cost(f:x") = c(f).

1) Thus, cost(f:x') = cost(f:x").

case 2) If f=G(f1,...,fn) then all the canonical restrictions of f are

given by one of the cases of definition 3.5.A, for some f_i,

i=1,...,flast. By Theorem 3.3 d($f_j{}^C$) $\subseteq$ d(f_i), for all $f_j{}^C$,

j=1,...,p(f). Thus, if x',x" are in d($f_j{}^C$) for some j then x',x" are in

75

$d(f_i)$ for one and only one i, i=1,...,flast. Then by Definition 3.1.B

and Definition 3.3, $P_i(f,x')$=True and $P_i(f,x'')$=True for one and only one

i, and cost(f:x') and cost(f:x'') are given by the same cost function

under Definition 4.1. For f:x=$\overrightarrow{y}$:x, it is direct that cost(f:x') = cost

(f:x'') for all x',x'' in $d(f_i)$. For the cases where f_i is not a constant

function a proof by mathematical induction is given.

Let S(N) be the statement, "If f=G(f1,...,fn) and f is defined by

at most N applications of the expansion rules of G, then if x',x'' are in

$d(f_i^c)$ for any f_i^c, i=1,...,p(f), cost(f:x') = cost(f:x'')."

Basis: If N=1 and f=G(f1,...,fn) then f1,...,fn are primitive functions

in f. Proof that S(1) is true is given for each case of f_i in

Definition 4.1.

b) Suppose f_i^c:x=$f1_j^c$:($f2_k^c$:x)and x',x'' $\in$ $d(f_i^c)$.

  ==> x, x' $\in$ $d(f2_k^c)$ & $f2_k^c$:x, $f2_k^c$:x' $\in$ $d(f1_1^c)$ Df. 3.5.B

  ==> x',x'' $\in$ {x| $P_1(f,x)$ & $P_k(f2,x)$ & $P_j(f1,x)$} Df. 3.5.A

  ==> $P_1(f,x')$ & $P_1(f,x'')$ & cost(f2:x')=cost(f2:x'')

    & cost(f1:(f2:x'))=cost:(f1:(f2:x'' (1) Df. 4.1

Then by Definition 4.1

  cost(f:x') = c($\oplus$) + cost(f2:x') + cost(f1:(f2:x'))

By (1) cost(f2:x')=cost(f2:x'') and cost(f1:(f2:x'))=cost(f1:(f2:x'')).

Thus, cost(f:x') = c($\oplus$) + cost(f2:x'') + cost(f1:(f2:x'')).

  = cost(f1:(f2:(x'')))

  = cost(f:x'')

Thus, cost(f:x') = cost(f:x'')

c) Let f:x=[f1,...,fn]:x. The proofs of both cases of f_i, i=1,2 are

identical.

Let x',x'' $\in$ $d([f1_j^c,...,fn_k^c]_1^c)$, for some (j,k).

==> x', x" $\in$ (d(f1$_j^c$) $\cap$...$\cap$ d(fn$_k^c$))   Df. 3.5.B

==> x',x" $\in$ {x| P_1(f,x) & P_j(f1,x) &...& P_k(fn,x)}

                                             Df. 3.5.A

==> P_1(f,x') & P_1(f,x") & (cost(f1:x')=cost(f1:x"))..

      &...(cost(fn:x')=cost(fn:x"))   (2)   Df. 4.1

==> cost(f1:x) +...+ cost(fn:x') = cost(f1:x") +...+ cost(fn:x")

Then by Definition 4.1

     cost(f:x') = c([]) + cost(f1:x') +....+ cost(fn:x')

By equation (2) above

    = c([]) + cost(f1:x") +...+ cost(fn:x")

    = cost([f1,...,fn]:x"

    = cost(f:x")

Thus, cost(f:x') = cost(f:x")

d) Let f:x = (f1-->f2;f3):x. The proofs for all 3 cases of f$_i$,

i=1,...,3 are identical. They are similar to case c.

Inductive Step. Let S(N) be true for all N $\geq$ 1. Then if f=G(f1,...,fn)

and f is defined by at most N+1 applications of the expansion rules of

G, f1,...,fn are defined by at most M $\leq$ N applications of the expansion

rules of G. By the inductive assumption, if x',x" $\in$ d(f1$_i^c$), for some

i=1,...,p(f1), cost(f1:x') = cost(f1:x"). The same holds for f2,...,fn.

Proof that S(N+1) is true is given below for the various cases of f.

    b) Let f:x=(f1$\oplus$f2):x and let x',x" $\in$ d((f1$_i^c\oplus$f2$_j^c$)$_1^c$:x), for some

(i,j).

     ==> x',x" $\in$ d(f2$_j^c$)   Df. 3.5.B

Then by the inductive assumption for f2,

     cost(f2:x') = cost(f2:x").   (3)

Also, x',x" $\in$ d((f1$_i^c\oplus$f$_2$j$\oplus$c)$_1^c$)

$$\Rightarrow (f2_j{}^C{:}x' \in d(f1_i{}^C)) \ \& \ (f2_j{}^C{:}x'' \in d(f1_i{}^C)) \qquad Df. \ 3.5.B$$

Then by the inductive assumption for f1,

$$cost(f1{:}(f2{:}x')) = cost(f1{:}(f2{:}x'')) \qquad (4) \qquad Df. \ 4.1$$

Then by Definition 4.1

$$cost(f{:}x') = c(\oplus) + cost(f2{:}x') + cost(f1{:}(f2{:}x'))$$

By equations (3) and (4) above,

$$= c(\oplus) + cost(f2{:}x'') + cost(f1{:}(f2{:}x''))$$

$$= cost((f1\oplus f2){:}x'')$$

$$= cost(f{:}x'')$$

b) Thus, $cost(f{:}x') = cost(f{:}x'')$

c,d) Cases c and d are similar to b.

2) Thus, If $S(N)$ Is true then $S(N+1)$ is true for all $N \geq 1$.

Therefore, $S(N)$ is true for all $N \geq 1$.

Therefore, for all f in F If $x',x'' \in d(f_i{}^C)$, for some $i=1..p(f)$ then $cost(f{:}x') = cost(f{:}x'')$.

APPENDIX G

COMPUTED RESULTS

The results in this table were computed by the program described in the introduction of this paper except for the case denoted by an '*' which were computed by hand.

1) $f:x = ((atom \oplus s) \to s; tl):x$

   a) $f_1^C = ((atom_2^C \oplus s_1^C) \to s; tl_1^C)_2^C$

      $d(f_1^C) = \langle U_i \langle Z^i(X.1.i) \rangle (X.1) \rangle (X)$

      $r(f_1^C) = \langle \rangle$

      $c(f_1^C) = c(\to) + c(\oplus) + c(s) + c(atom) + c(*)$

   b) $f_1^C = ((atom_2^C \oplus s_1^C) \to s; tl_2^C)_2^C$

      $d(f_1^C) = U_i \langle U_j \langle Z^j(X.1.j) \rangle (X.1), Z^i(X.i+1) \rangle (X)$

      $r(f_1^C) = U_i \langle Z^i(X.i+1) \rangle (|X|-1)$

      $c(f_1^C) = c(\to) + c(\oplus) + c(s) + c(atom) + c(tl)$

   c) $f_1^C = ((atom_1^C \oplus s_1^C) \to s_1^C; tl)_1^C$

      $d(f_1^C) = U_i \langle Atom(X.1), Z^i(X.i+1) \rangle (X)$

      $r(f_1^C) = Atom(X.1)$

      $c(f_1^C) = c(\to) + c(\oplus) + c(s) + c(atom) + c(s)$

             $= c(\to) + c(\oplus) + 2c(s) + c(atom)$

d)* $f_1{}^c$ = $((atom_3{}^c \oplus s_2{}^c)\text{-->}s; \, tl)_3{}^c$

$U_1(d(f_1{}^c))$ = Atoms $U \perp$

$d(f_1{}^c)$ = Atoms $U \perp$

$r(f_1{}^c)$ = $\perp$

$c(f_1{}^c)$ = $c(\text{-->}) + c(\oplus) + c(\#) + c(atom)$

2) $f:x = ((\geq \oplus[s,\overline{5}])\text{-->}tl; \, rev):x$

a) $f_1{}^c$ = $((\geq_1{}^c \oplus [s_1{}^c, \overline{5}_1{}^c]_1{}^c)\text{-->}tl_2{}^c; \, rev)_1{}^c$

$d(f_1{}^c)$ = $U_1 <NUM(X.1), \, Z^1(X.i+1)>(X), \, (X.1 \geq 5)$

$r(f_1{}^c)$ = $U_1 <Z^1(X.i+1)>(|X|-1)$

$c(f_1{}^c)$ = $c(\text{-->}) + c(\oplus) + c([]) + c(s) + c(\#) + c(\geq) + c(tl)$

b) $f_1{}^c$ = $((\geq_1{}^c \oplus [s_1{}^c, \overline{5}_1{}^c]_1{}^c)\text{-->}tl_1{}^c; \, rev)_1{}^c$

$d(f_1{}^c)$ = $<NUM(X.1)>(X), \, (X.1 \geq \overline{5})$

$r(f_1{}^c)$ = $<>$

$c(f_1{}^c)$ = $c(\text{-->}) + c(\oplus) + c([]) + c(s) + c(\#) + c(\geq) + c(\#)$

= $c(\text{-->}) + c(\oplus) + c([]) + c(s) + 2c(\#) + c(\geq)$

c) $f_1{}^c$ = $((\geq_2{}^c \oplus [s_1{}^c, \overline{5}_1{}^c]_1{}^c)\text{-->}tl; \, rev_2{}^c)_2{}^c$

$d(f_1{}^c)$ = $U_1 <NUM,(X.1), \, Z^1(X.i+1)>(X), \, (X.1 < 5)$

$r(f_1{}^c)$ = $U_1 <Z^1(X.i+1), \, NUM(X.1)>(X)$

$c(f_1{}^c)$ = $c(\text{-->}) + c(\oplus) + c([]) + c(s) + c(\#) + c(\geq) + c(rv)$

d) $f_1{}^c$ = $((\geq_2{}^c \oplus [s_1{}^c, \overline{5}_1{}^c)\text{-->}tl; \, rev_1{}^c)_2{}^c$

$d(f_1{}^c)$ = $<NUM(X.1)>(X), \, (X.1 < 5)$

$r(f_1{}^c)$ = $<NUM(X.1)>(X)$

$c(f_1{}^c)$ = $c(\text{-->}) + c(\oplus) + c([]) + c(s) + c(\#) + c(\geq) + c(\#)$

= $c(\text{-->}) + c(\oplus) + c([]) + c(s) + 2(c(\#) + c(\geq))$

e)* f_1^C $= ((\geq_3^C \oplus [s_2^C, \bar{5}_1^C]_2^C(T,F)) \text{---}>tl; \ rev)_3^C$

$U_1 d(f_1^C)$ = Atoms $U \perp$

$d(f_1^C)$ = Atoms

$r(f_1^C) = \perp$

$\max(c(f_1^C)) = c(\text{---}>) + c(\oplus) + c([]) + 2c(\#) + c(\geq)$

3) $f:x = (apndl \oplus ([\bar{9}, id] \oplus tl)):x$

a) f_1^C $= (apndl_1^C \oplus ([9_1^C, id_1^C]_1^C \oplus tl_1^C)$

$d(f_1^C) = <Z(X1)>(X)$

$r(f_1^C) = <9>$

$c(f_1^C) = c(\oplus) + c(\oplus) + c(\#) + c([]) + c(id) + c(\#)$

$\qquad + c(apndl1)$

$\qquad = 2c(\oplus) + 2c(\#) + c([]) + c(id) + c(apndl1)$

b) f_1^C $= (apndl_2^C \oplus ([\bar{9}_1^C, id_1^C]_1^C \oplus tl_2^C)$

$d(f_1^C) = U_1 <Z(X.1), \ Z^1(X.i+1)>(X)$

$r(f_1^C) = U_1 <9, \ Z^1(X.i+1)>(X)$

$c(f_1^C) = c(\oplus) + c(\oplus) + c(tl) + c([]) + c(id) + c(\#)$

$\qquad + c(apndl2)$

$\qquad = 2c(\oplus) + c(tl) + c([]) + c(id) + c(\#)$

$\qquad + c(apndl2)$

c)* f_1^C $= (apndl_3^C([\bar{9}_2^C, id_1^C]_2^C(T,T) \oplus tl_3^C)$

$U_1 d(f_1^C)$ = Atoms $U \perp$

$d(f_1^C) = \perp$

$r(f_1^C) = \perp$

$\max(c(f_1^C)) = 2c(\oplus) + 3c(\#) + c([]) + c(id)$

4) $f:x = (apndl \oplus ([\bar{9}, tl]):x$

   a) $f_|{}^c$ $(apndl_1{}^c \oplus [\bar{9}_1{}^c, tl_1{}^c]_1{}^c)$

     $d(f_|{}^c)$ = same as 3.a

     $r(f_|{}^c)$ = same as 3.a

     $c(f_|{}^c) = c(\oplus) + c([]) + 2c(\#) + c(apndl1)$

   b) $f_|{}^c$ $= (apndl_2{}^c \oplus [\bar{9}_1{}^c, tl_2{}^c]_1{}^c)$

     $d(f_|{}^c)$ = same as 3.b

     $r(f_|{}^c)$ = same as 3.b

     $c(f_|{}^c) = c(\oplus) + c([]) + c(tl) + c(\#) + c(apndl2)$

   c)* $f_|{}^c$ $= (apndl_3{}^c \oplus [\bar{9}_2{}^c, tl_3{}^c]_2{}^c(T,T))$

     $U_|d(f_|{}^c)$ = same as 3.c

     $d(f_|{}^c)$ = same as 3.c

     $r(f_|{}^c)$ = same as 3.c

     $max(c(f_|{}^c)) = c(\oplus) + c([]) + 3c(\#)$

5) $f:x = (apndl \oplus [atom \oplus tl, [+\oplus tl, *\oplus tl]]):x$

   a) $f_|{}^c = (apndl_2{}^c \oplus [atom_2{}^c \oplus tl_2{}^c, [+_1{}^c \oplus tl_2{}^c, *_1{}^c \oplus tl_2{}^c]_1{}^c]_1{}^c$

     $d(f_|{}^c) = <Z(X.1), NUM(X.2), NUM(X>2)>(X)$

     $r(f_|{}^c) = <F, NUM(X.2+X.3), NUM(X.2*X.3)>$

     $c(f_|{}^c) = 4c(\oplus) + 2c([]) + 3c(tl) + c(*) + c(+) + c(atom)$

            $+ c(apndl)$

   b)* $f_|{}^c$

     $= (apndl_3{}^c \oplus [atom_2{}^c \oplus tl_2{}^c, [+_2{}^c \oplus tl_2{}^c, *_2{}^c \oplus tl_2{}^c]_2{}^c(T,T)]_2{}^c(T,T)$

     $U_|d(f_|{}^c) = Atoms \ U \ \underline{|} \ U \ <Z> \ U \ <Z,Z> \ U \ <Z,Z-NUM,Z-NUM>$

           $U \ <Z,Z,Z-NUM> \ U \ <Z,Z-NUM,Z> \ U_|<Z,Z,Z,Z^|>$

     $d(f_|{}^c) = <Z,Z> \ U \ <Z,Z-NUM,Z-NUM> \ U \ <Z,Z-NUM,Z>$

$$U <Z,Z,Z-NUM> \quad U_|<Z,Z,Z,Z^|>$$

$r(f_|^C) = \underline{\bot}$

$max(c(f_|^C)) = 4c((\oplus) + 2c([]) + 3c(t|) + 3c(\#) + c(atom)$

6) $f:x = [atom\oplus t|, +\oplus t|, *\oplus t|]:x$

   a) $f_|^C = [atom_2{}^C\oplus t|_2{}^C, +_1{}^C\oplus t|_2{}^C, *_1{}^C\oplus t|_2{}^C]_1{}^C$

      $d(f_|^C) = $ same as 5.a

      $r(f_|^C) = $ same as 5.a

      $c(f_|^C) = c([]) + 3c(\oplus) + 3c(t|) + c(*) + c(+) + c(atom)$

   b)* $f_|^C = [atom_2{}^C\oplus t|\oplus 2^C, +_2{}^C\oplus t|_2{}^C, *_2{}^C\oplus t|_2{}^C]_2{}^C(T,T,T)$

      $U_|d(f_|^C) = $ same as 5b

      $d(f_|^C) = $ same as 5b

      $r(f_|^C) = $ same as 5b

      $max(c(f_|^C)) = c([]) + 3c(\oplus) + 3c(t|) + 2c(\#) + c(atom)$

7) $f:x = [atom, + , *]:x$

   a) $f_|^C = [atom_2{}^C, +_1{}^C, *_1{}^C]_1{}^C$

      $d(f_|^C) = $ same as 5.a and 6.a

      $r(f_|^C) = $ same as 5.a and 6.a

      $c(f_|^C) = c(\oplus) + c(t|) + c([]) + c(*) + c(+) + c(atom)$

   b)* $f_|^C = [atom_2{}^C, +_2{}^C, *_2{}^C]_2{}^C\oplus t|_2{}^C(T,T,T)$

      $U_|d(f_|^C) = $ same as 5.b and 6.b

      $d(f_|^C) = $ same as 5.b and 6.b

      $r(f_|^C) = $ same as 5.b and 6.b

      $max(c(f_|^C)) = c(\oplus) + c(t|) + c([]) + 2c(\#) + c(atom)$

VITA $\mathcal{2}$

Richard Walter Matzen

Candidate for the Degree of

Master of Science

Thesis: A CHARACTERIZATION OF FUNCTIONS FOR EXECUTION TIME COST ANALYSIS IN FP

Major Field: Computing and Information Science

Biographical:

Personal Data: Born in Rochester, New York, May 12, 1948, the son of Walter T. and Virginia Matzen. Married to Janet E. Chaney. on February 28, 1974.

Education: Graduated from Richardson High School, Richardson Texas, in May, 1966; received Bachelor of Science Degree from the University of Central Arkansas in August, 1984; completed requirements for the Master of Science Degree at Oklahoma State University in July, 1987.

Professional Experience: Teaching Assistant, Department of Computing and Information Sciences, Oklahoma State University, August, 1984, to August, 1986; Lecturer, Department of Computing and Information Sciences, Oklahoma State University, August, 1986, to January, 1987; Software Engineer, Time Management Software, Inc., Stillwater, Oklahoma, February, 1987, to present.