TRANSLATION NETWORKS AND FUNCTION COMPOSITION

Peter Wegner

Technical Report

No. 68-10

March 1968

Department of Computer Science
Cornell University
Ithaca, New York 14850

Translation Networks and Function Composition

Peter Wegner

Cornell University

March 1968

Introduction

In reference 1, Slansky, Finkelstein and Russell develop a formalism for specifying program translation functions in terms of their domains, ranges and representation. They introduce a "cancellation law" which allows the effect of application of a sequence of translation functions to be specified in terms of their notation, and they prove that parentheses which specify the order of function application can be omitted and subsequently recovered. However, the reasons which make their formalism work are not clearly stated in their paper. Their discussion makes it appear that they have invented a mysterious trick, and that they are pulling rabbits out of a hat when they simplify sequences of translation functions by applying the cancellation law. In the present paper the algebraic basis of these results will be developed. It will be shown that the dropping of parentheses is a direct consequence of the associativity of function composition, and that the cancellation laws are a consequence of cancellation of intermediate domains and ranges during function composition.

Slansky, Finkelstein and Russell begin by introducing the notions of algorithm, language, program, data collection, translation algorithm, translator, execution expression and expression, and show how these concepts are

interrelated. We shall start with the notion of a function, and show that other notions that arise in specifying program translation may be regarded as special cases of the notion of a function.

Domains, Ranges and Function Composition

A function f may be defined as a rule of correspondence between a domain D and a range R . When considering functions which accomplish program translation we are frequently more interested in the domain and range of a function than in the precise rule of correspondence between the domain and the range. Following (1) we shall denote by D/R any function having domain D and range R . When the function is a translation program the domain D is the source language and the range R is the target language.

We shall be concerned with functions defined in terms of the composition of other functions. Let $f = f_3 f_2 f_1$ be the function which the the result of first applying f_1 to an argument x in its domain, then applying f_2 to the element $f_1(x)$, and then applying f_3 to $f_2(f_1(x))$. Let $f_1 = A/B$, $f_2 = C/D$ and $f_3 = E/F$. Then $f = \frac{E}{F} \frac{C}{D} \frac{A}{B}$ is a function which has a domain A and is defined for all elements of its domain if and only if the range B of f_1 is a subset of the domain C of f_2 and if the range D of f_2 is a subset of the domain E of f_3 . When $C = B$ and $E = D$ then f has the domain A and the range F . In this case we can write $f = \frac{D}{F} \frac{B}{D} \frac{A}{B} = \frac{A}{F}$. i.e. If the domain of f_2 equals the range of f_1 and the domain of f_3 equals the range of f_2 then $f = f_3 f_2 f_1$ is a function whose domain is

that of f_1 and whose range is that of f_3 . More generally if $f = f_k \dots f_2 f_1$ and the range of f_i is the domain of f_{i+1} for $i = 1, 2, \dots, k-1$, then domains and ranges can be cancelled by ordinary cancellation laws, so that f has the domain of f_1 and the range of f_k .

Although we normally think of sequences of functions such as $f_3 f_2 f_1$ as not being applicable until an argument x is supplied, there are certain functions which expect functions as their arguments. If f_2 and f_3 are functions which expect functions as their arguments, then $f_2(f_1)$, $f_3(f_2)$, $f_3(f_2(f_1))$ and $f_3(f_2)(f_1)$ may be evaluated, and the statement $\frac{D}{F} \frac{B}{D} \frac{A}{B} \rightarrow \frac{A}{F}$ may be interpreted dynamically as application of f_2 to f_1 followed by application of f_3 to $f_2(f_1)$. Translators are examples of functions which expect functions (programs) as their arguments. Slansky, Finkelstein and Russell use the above notation for function composition to specify the application of translators to programs.

If $f = f_k \dots f_2 f_1$ and each f_i represents a translation program then the condition that $R(f_i) = D(f_{i+1})$ becomes the condition that the target language of f_i is the source language of f_{i+1} .

It is well known that function composition is associative. Thus $f_k \dots f_2 f_1$ may be grouped in any way which preserves the consistency of domains and ranges, and still lead to the same result.

Representation and Interpretation

When a function is applied to its arguments it must be represented as a program in some programming language, and executed by some interpreter. Let rep_L be a function which converts a function f into a function $\text{rep}_L(f)$ in the programming language L , and let ap_L be the function which applies programs in the language L to their data. The function ap_L has a domain L and will therefore be denoted by L . The function rep_L has a range L and will therefore be denoted by L/L .^{*} A function f in the language L will be denoted by $L/L f = \frac{f}{L}$. If f has a domain A and a range B , then it will be denoted by $\frac{A}{LB}$.

A data item may be thought of as a function which has a range without having a domain. Thus the data set of a function with domain D may be specified as L/D .

The above notions will now be used to recast one of the examples of Slansky, Finkelstein and Russell in function terminology. The change of terminology indicates that the characteristics of translation networks can

* Note that ap_L is a left inverse of rep_L in the sense that $\text{ap}_L(\text{rep}_L(f(x)))$ produces the value $f(x)$ which results from applying f to x . The mapping rep_L operating on the function with its data, followed by the interpreter ap_L acting on the function and data in the representation L , makes explicit the fact that mechanical evaluation requires a function with its data to be represented in a particular representation, and executed (interpreted) in that representation.

be specified explicitly in terms of domains and ranges of functions, and makes the notations of Slansky, Finkelstein and Russell seem natural rather than mysterious.

An assembler which translates from language L to language M and is written in M is represented by the function $\frac{L}{MM}$. A function q written in the language L is represented by $\frac{q}{L}$. The translation of the program q/L into the program q/M by the assembler $\frac{L}{MM}$ on the machine M is represented by $M \frac{L}{MM} \frac{q}{L} = \frac{q}{M}$. Note that the replacement of a sequence of functions by a single function is here identified with a computation which produces the function q/M .

Application of the program $\frac{q}{M}$ to its data δ on the machine M is represented by $M \frac{q}{M} \delta = q\delta$. Here the left hand side indicates explicitly the functions involved during execution and the right hand side indicates the result.

The combined translation and execution process can be specified as $M M \frac{L}{MM} \frac{q}{L} \delta = q\delta$. Here the left hand side specifies two phases of execution corresponding to the two instances of M . The associativity of function composition for functions with compatible domains and ranges allows us to group function constituents in any order and to associate a meaningful function with any of the partial groupings obtained.

The expression $MM \frac{L}{MM} \frac{q}{L} \delta$ contains two interpreters. The second

interpreter specifies application of $\frac{L}{MM}$ to $\frac{q}{L}$ producing $\frac{q}{M}$, and the first interpreter specifies application of $\frac{q}{M}$ to δ producing the result $q\delta$. Since function application in this notation can be specified only by an interpreter, every expression to be evaluated must contain $2n + 1$ components of which n components are interpreters, specifying application of the remaining $n + 1$ function components to each other.

-
1. J. Slansky, M. Finkelstein and E. C. Russell, "A Formalism for Program translation", JACM, April 1968.