

ARRAY THEORY AND THE NIAL PROGRAMMING LANGUAGE

K. W. SMILLIE

Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada T6G 2H1

Dedicated to Peter Naur on the occasion of his 60th birthday.

Abstract.

The Nested Interactive Array Language Nial is based on a mathematical theory of hierarchical rectangular arrays known as array theory. This paper gives a brief introduction to array theory and Nial and then discusses the main characteristics of the Nial language, its current implementation and some of its applications.

CR Subject Classifications: D.3.2, E.1, J.0.

Arrays as commonplace objects.

Arrays with one, two and three axes are objects with which we have always been familiar. Young children arrange blocks in more or less rectangular arrangements; they sit in school at desks arranged in rows in front of a blackboard which may be ruled in rows and columns. In the supermarket we see row upon row of shelves containing a multitude of goods arranged in a variety of arrays. Eggs are arranged in cartons with two rows of six eggs each, and have been taken from cartons in which they were placed in five layers of three boxes each. Apples come in boxes of four layers each with twenty-four apples arranged in four rows of six apples. Furthermore, an empty box of egg cartons retains its identity as an empty box of egg cartons and is easily distinguishable from an empty box of apples. Indeed, arrays are very much a part of our lives, and we spend much of our time creating, rearranging, moving, emptying and describing them.

Array theory has been developed over the past twenty years by Trenchard More of the IBM Cambridge Scientific Center and his colleagues [7, 8, 9]. It provides a coherent mathematical treatment of a universe consisting solely of nested rectangular arrangements of objects. A large number of operations with arrays as arguments have been defined as well as transformers which have these

operations as their arguments. Nial, from the Old Norse Icelandic name *Njal*, is a high-level programming language derived from array theory which combines some of the syntactic and semantic constructs of APL, Lisp and Pascal with operations for accessing files, providing graphics and managing workspaces in an interactive environment.

In this discussion of array theory and Nial only a few of the many operations and transformers have been introduced. For convenience these are given, each with a brief descriptive note, in Figure 1.

A mod B		Remainder of A upon division by B
A plus B	+	Sum of A and B
prod A	*	Product of the items of A
sum A	+	Sum of the items of A
A times B	*	Product of A and B
A above B	>	A greater than B
A equal B	=	A equal to B
count A		Integers 1 to A
first A		First item of A
front A		All items of A except the last
last A		Last item of A
rest A		All items of A except the first
A sublist B		Items of B selected by mask A
A append B		Items of A followed by B as last item
A hitch B		A as first item followed by items of B
A link B		Items of A followed by items of B
A reshape B		Items of B with shape A
empty A		Is A empty
pass A		A
void A		Empty list with archetype A
A CONVERSE f A B f A		
EACH f A		Apply f to items of A
A EACHRIGHT f B		Apply f to items of B with A fixed
FORK[f, g, h] A		If f A is Truth then g A else h A
A OUTER f B		Apply f to cartesian product of A and B
SORT f A		Sort A with ordering determined by f

Fig. 1. Primitive Nial operations and transformers.

Array theory.

The two principal concepts of array theory are the hierarchical nesting and the rectangular arrangement of objects which are considered fundamental for the mathematical description and analysis of many natural and man-made systems. Array theory is similar to set theory in that it is one-sorted and closed so that the only objects considered are nested arrays in which the nesting always terminates in objects which are arrays and contain themselves. Unlike set theory the objects of array theory occur in ordered rectangular arrangements of arbitrary finite valence. The operations of array theory hold for all arrays