A Characterization of Associativity

Arthur Charlesworth

IPC-TR-90-007
November 30, 1990

Institute for Parallel Computation
School of Engineering and Applied Science
University of Virginia
Charlottesville, VA 22903

and

Department of Mathematics and Computer Science
University of Richmond
Richmond, VA 23173

This research was supported by the National Aeronautics and Space Administration, under grant number NAG-1-774, by the Jet Propulsion Laboratory, under grant number 95722, and by sabbatical funding from the University of Richmond.

A Characterization of Associativity

Arthur Charlesworth
University of Richmond and University of Virginia

ABSTRACT: A necessary and sufficient condition for the associativity of a function is given, in terms of a particular relation being a function. The concept of an associative function is generalized to the concept of a function being associative relative to a sequence and a characterization of such relative associativity is also given. These two characterizations are applied to the problem of proving the associativity, or relative associativity, of a function.

Categories and Subject Descriptors: D.1.3 [Programming Techniques]: Concurrent Programming; D.3.3 [Programming Languages]: Language Constructs

General Terms: Languages

Additional Key Words and Phrases: associativity, reduction, parallel programming.

Contents

1 INTRODUCTION

Among the most efficiently implementable operations involving the participation of multiple processes are the associative operations. Partial computations of associative operations can be combined using a variety of efficient techniques, such as read/modify/write on shared memory machines, processor trees on distributed memory hypercubes, and shifts or pointer doubling on Single Instruction Multiple Data parallel computers. For this reason, support for computing reductions of finite sequences using standard associative functions, such as +, *, `and`, `or`, `max`, and `min`, is commonly provided within languages for parallel computing.[1]

Support for using less trivial programmer-defined associative functions is provided by a *general reduction operator*. Given that a function f is known to be associative, a general reduction operator $\mathcal{R}$ obtains the reduction $\mathcal{R}(f, s)$ of a finite sequence s using the function f. Such a general reduction operator is provided in several languages for parallel computing, such as iPSC/2 Fortran and C [Int89] and the innovative and less conventional languages Connection Machine Lisp [HS86] and Paralation Lisp [Sab88]. Since $\mathcal{R}$ need apply f only to the sequence s, it is not necessary that f be associative, only that f satisfy an associativity property relative to the sequence s. The concept of associativity relative to a sequence is defined and studied in this paper. This concept extends the applicability of $\mathcal{R}$; e.g., there exists a nonassociative function that is associative relative to any nondecreasing sequence of records.

It is not always easy to prove the associativity of a function f, defined by a function program written in a programming language. To illustrate this, notice that since the defining equation $f(x, f(y, z)) = f(f(x, y), z)$ for associativity involves 4 applications of f, a straightforward proof of the associativity of f using such a proof of correctness involves the consideration of p^4 cases, where p is the number of cases used in proving the correctness of a single application of f. A similar observation can be made about proving relative associativity. Although ad-hoc arguments based on special properties of f can reduce the number of cases significantly, the effort required to reduce the number of cases can increase the difficulty of the associativity proof. The main results of this paper are characterization theorems for associativity and relative associativity, which provide an alternative approach to proving associativity that requires the consideration of no more than p cases.

[1] Language support for reductions was provided much earlier by APL [Ive62].

2 ASSOCIATIVITY RELATIVE TO A SEQUENCE

Before presenting a mathematical characterization of associativity it is necessary to provide preliminary definitions and results related to associativity and reductions. Throughout this paper the notation $< s_1, \cdots, s_n >$ denotes a sequence s of n elements and $\circ$ denotes string concatenation. By a "slice" of a sequence we shall mean a sequence of consecutive terms of the given sequence.

We begin by defining the concept of associativity relative to a sequence. Intuitively, a function is associative relative to a sequence if the defining equation for associativity $f(x, f(y, z)) = f(f(x, y), z)$ is satisfied whenever x, y, and z result from applying f zero or more times, grouping from the right, to three adjacent nonnull finite slices of the given sequence. To facilitate a more precise definition of relative associativity, we define the f-reduction of a sequence, even if the function f fails to be associative.

DEFINITION 1. Let s be a finite sequence of elements of E and let f be a function from $E \times E$ to E. The *f-reduction* of the sequence $< x > \circ s$, denoted $f_r(< x > \circ s)$, is defined to equal x if s is the null string and to equal $f(x, f_r(s))$ otherwise.

DEFINITION 2. Let s be a sequence of elements of E and let f be a function from $E \times E$ to E. To say that f is *associative relative to* s means that $f(f_r(s_1), f(f_r(s_2), f_r(s_3))) = f(f(f_r(s_1), f_r(s_2)), f_r(s_3))$ holds for any nonnull finite sequences s_1, s_2, and s_3 such that $s_1 \circ s_2 \circ s_3$ is a slice of s.

Notation for slices of a sequence is used far more often than notation for terms of a sequence in this paper. Thus for convenience the subscript notation s_i, when used outside angle brackets, denotes a slice of s. When it is necessary to denote a term of s outside angle brackets, a phrase such as "the i^{th} term of s" will be used.

LEMMA 1. *Let f be a function from $E \times E$ to E and let s be a sequence of elements of E. If f is associative relative to s, then $f(x, f(y, z)) = f(f(x, y), z)$ holds for each triple of consecutive terms x, y, and z of s.*

PROOF. Let s_1, s_2, and s_3 be $< x >$, $< y >$, and $< z >$, respectively. Then

$$\begin{aligned} f(x, f(y, z)) &= f(f_r(< x >), f(f_r(< y >), f_r(< z >))) \\ &= f(f(f_r(< x >), f_r(< y >)), f_r(< z >)) \\ &= f(f(x, y), z) \end{aligned}$$

where the first and last equality follow from the definition of f_r and the second equality follows from the fact that f is associative relative to s. □

The following example shows that the converse of Lemma 1 does not hold and thus the notion of relative associativity would be weakened if Definition 2 were phrased more directly in terms of the defining equation for associativity.

EXAMPLE 1. *There is a function f from $E \times E$ to E and a sequence s of elements of E such that $f(x, f(y,z)) = f(f(x,y),z)$ for each triple of consecutive terms x, y, and z of s, yet f fails to be associative relative to s.* For let $E = \{1,2,3,4\}$, let s be the sequence $< 1,2,3,4 >$, and let f be any function from $E \times E$ to E such that

$$\begin{aligned} f(1,4) &= f(2,3) = f(2,4) = 1 \\ f(1,1) &= f(3,3) = 2 \\ f(1,2) &= 3 \\ f(3,4) &= 4 \end{aligned}$$

Then

$$f(1, f(2,3)) = f(f(1,2),3)$$

since both sides evaluate to 2, and

$$f(2, f(3,4)) = f(f(2,3),4)$$

since both sides evaluate to 1. However f is not associative relative to s since, if $s_1 =< 1,2 >$, $s_2 =< 3 >$, and $s_3 =< 4 >$, then

$$\begin{aligned} f(f_r(s_1), f(f_r(s_2), f_r(s_3))) &= f(f(1,2), f(3,4)) \\ &= f(3,4) \\ &= 4 \end{aligned}$$

whereas

$$\begin{aligned} f(f(f_r(s_1), f_r(s_2)), f_r(s_3)) &= f(f(f(1,2),3),4) \\ &= f(f(3,3),4) \\ &= f(2,4) \\ &= 1. \square \end{aligned}$$

The definition of associativity relative to a sequence is based on the arbitrary choice of defining f_r so that grouping is from the right. The next result demonstrates that, in fact, the choice of grouping does not matter. To simplify the statement of this result, for a function f from $E \times E$ to E and a sequence s of n elements of E, we use the phrase "reduct of s with respect to f" to mean any element of E that can be obtained by applying f a total of $n-1$ times

to the sequence, according to the ordering imposed by the sequence but with no restriction on the grouping used. More precisely: for a sequence having a single term, the term itself is a reduct of the sequence with respect to f; for a sequence $< x, y >$ of two terms, the element $f(x, y)$ of E is a reduct of the sequence with respect to f, and for any other sequence $< x > o t o < y >$, the elements $f(a, y)$ and $f(x, b)$ of E are reducts of the sequence with respect to f, where a is a reduct of $< x > o t$ with respect to f and b is a reduct of $t o < y >$ with respect to f. There are no other reducts of a sequence with respect to f beyond those just specified.

LEMMA 2. *Let f be a function from $E \times E$ to E and let s be a sequence of elements of E. Then f is associative relative to s if and only if for any nonnull finite sequences s_1, s_2, and s_3 such that $s_1 \circ s_2 \circ s_3$ is a slice of s, $f(\overline{s_1}, f(\overline{s_2}, \overline{s_3})) = f(f(\overline{s_1}, \overline{s_2}), \overline{s_3})$, where for $i = 1, 2, 3$, $\overline{s_i}$ denotes a reduct of s_i with respect to f.*

PROOF. Clearly f is associative relative to s if the stated condition holds, since for any sequence u, $f_r(u)$ is easily seen to be a reduct of u. To see the converse, assume that f is associative relative to s. It suffices to show that for each slice t of s, $f_r(t)$ is equal to each reduct of t with respect to f, and this can be proven by strong induction on the number of terms of t as follows. The result clearly follows from the definition of f_r if t has 1 or 2 terms, and the result follows from Lemma 1 if t has 3 terms, so assume the result holds for slices shorter than t, where t has more than 3 terms. Let x and y be the first and last terms of t so that $t =< x > o u o < y >$, for some slice u of s of length at least 2, say $u =< z > o v$. Each reduct of t has either the form $f(a, y)$ or the form $f(x, b)$, where a is a reduct of $< x > o u$ with respect to f and b is a reduct of $u o < y >$ with respect to f. Now

$$\begin{aligned} f(x, b) &= f(x, f_r(u o < y >)) \\ &= f_r(t) \end{aligned}$$

by the induction hypothesis and the definition of f_r, and

$$\begin{aligned} f(a, y) &= f(f_r(< x > o u), y) \\ &= f(f(x, f_r(u)), y) \\ &= f(f(f_r(< x >), f_r(u)), f_r(< y >)) \\ &= f(f_r(< x >), f(f_r(u), f_r(< y >))) \\ &= f(x, f(f_r(u), f_r(< y >))) \\ &= f(x, f(f_r(< z > o v), f_r(< y >))) \\ &= f(x, f(f(z, f_r(v)), f_r(< y >))) \\ &= f(x, f(f(f_r(< z >), f_r(v)), f_r(< y >))) \\ &= f(x, f(f_r(< z >), f(f_r(v), f_r(< y >)))) \end{aligned}$$

$$
\begin{aligned}
&= f(x, f(z, f(f_r(v), y))) \\
&= f(x, f(z, f_r(v \circ < y >))) \\
&= f(x, f_r(u \circ < y >)) \\
&= f_r(t)
\end{aligned}
$$

where the first equality and third from last equality follow from the induction hypothesis, the fourth and ninth equalities follow from the assumption that f is associative relative to s, the sixth equality follows from the definition of v, and the remaining equalities follow from the definition of f_r.□

LEMMA 3. *Let f be a function from $E \times E$ to E. The following statements are equivalent:*

1. *f is associative.*
2. *f is associative relative to s, for each sequence s of elements of E.*
3. *f is associative relative to s, for each nonnull finite sequence s of elements of E.*

PROOF. To see that (1) implies (2), let f be associative, let s be given, and let s_1, s_2, and s_3 be nonnull finite sequences such that $s_1 \circ s_2 \circ s_3$ is a slice of s. By denoting the elements $f_r(s_1)$, $f_r(s_2)$, and $f_r(s_3)$ of E by x, y, and z, respectively, and by applying the defining equation for associativity, it is clear that f is associative relative to s.

Since each nonnull finite sequence is a sequence, (2) implies (3).

To see that (3) implies (1), assume f is associative relative to s, for each nonnull finite sequence s of elements of E, and let x, y, and z be any elements of E. Since f is associative relative to the sequence $< x, y, z >$, it follows from Lemma 1 that $f(x, f(y, z)) = f(f(x, y), z)$. □

The following example illustrates the fact that there are naturally occurring functions for which associativity relative to a sequence is more general than associativity.

EXAMPLE 2. *There exists a nonassociative function that is associative relative to any nondecreasing sequence of records.* The maximal plateau problem [Gri81] asks for the length of the longest plateau in a nondecreasing sequence u of integers, where a plateau is a slice of the sequence all of whose values are equal. The maximal plateau problem can be solved using a general reduction operator as follows. Define a type **PLATEAU_TYPE** consisting of records having integer fields **LEN**, **FIRST_LEN**, **LAST_LEN**, **FIRST**, and **LAST**, and define a sequence s of such records such that the first three fields of each record of s have the value 1 and the remaining two fields of the i^{th} term of s have the value of the i^{th}

term of u. The ordering on u imposes a corresponding ordering on s so that s becomes a nondecreasing sequence of records. The length of the longest plateau of u can then be obtained by computing $\mathcal{R}(f, s)$, where $\mathcal{R}$ is a general reduction operator and f is the function given in Figure 1. The intent of the programmer is that the following specification be satisfied, when f is applied to a nonnull slice $t =< t_i, \ldots, t_j >$ of s, and where u_t denotes $< u_i, \ldots, u_j >$: **ANS.LEN** is the length of the longest plateau in u_t, where u_t is in nondecreasing order, **ANS.FIRST_LEN** is the length of the longest plateau in u_t that starts at the first component of u_t, **ANS.LAST_LEN** is the length of the longest plateau in u_t that ends at the last component of u_t, **ANS.FIRST** is the value of the first component of u_t, and **ANS.LAST** is the value of the last component of u_t.

To see the nonassociativity of f, let **X** and **Y** be records of type **PLATEAU_TYPE** such that

```
X.LEN       = 1        Y.LEN       = 1
X.FIRST_LEN = 1        Y.FIRST_LEN = 1
X.LAST_LEN  = 1        Y.LAST_LEN  = 1
X.FIRST     = 10       Y.FIRST     = 20
X.LAST      = 10       Y.LAST      = 20
```

Then `f(X, f(Y,Z))` differs from `f(f(X,Y), Z)`, where **Z** has the same value as **X**, since

```
f(X, f(Y,X)).LEN       = 1   f(f(X,Y), X).LEN       = 1
f(X, f(Y,X)).FIRST_LEN = 1   f(f(X,Y), X).FIRST_LEN = 2
f(X, f(Y,X)).LAST_LEN  = 2   f(f(X,Y), X).LAST_LEN  = 1
f(X, f(Y,X)).FIRST     = 10  f(f(X,Y), X).FIRST     = 10
f(X, f(Y,X)).LAST      = 10  f(f(X,Y), X).LAST      = 10
```

For an example in which the lack of associativity of f causes two different values to be computed for the field **LEN**, note that

```
f(X, f(X, f(Y,X)).LEN = 2
```

whereas

```
f(X, f(f(X,Y), X)).LEN = 3.
```

A proof of the relative associativity of f appears in Section 4.1.□

The function in Example 2 is also associative relative to the sequence naturally corresponding to any nonincreasing sequence and, more generally, any sequence in which each subsequence consisting of equal component values is a slice. (This is consistent with an observation in [Gri81] concerning a program to solve the maximal plateau problem.)

Other applications of using general reduction operators that, like Example 2, involve the use of records to collect information are discussed in [Cha90]. Among

```
function f (L, R: in PLATEAU_TYPE) return PLATEAU_TYPE is

  ANS: PLATEAU_TYPE;

  function MAX (X, Y: in INTEGER) return INTEGER is
  begin
    if X > Y then
      return X;
    else
      return Y;
    end if;
  end MAX;

begin -- f
    if L.LAST = R.FIRST then
      ANS.LEN := MAX (MAX (L.LEN, R.LEN), L.LAST_LEN + R.FIRST_LEN);
    else
      ANS.LEN := MAX (L.LEN, R.LEN);
    end if;
    if L.FIRST = R.FIRST then
      ANS.FIRST_LEN := L.LEN + R.FIRST_LEN;
    else
      ANS.FIRST_LEN := L.FIRST_LEN;
    end if;
    if L.LAST = R.LAST then
      ANS.LAST_LEN := L.LAST_LEN + R.LEN;
    else
      ANS.LAST_LEN := R.LAST_LEN;
    end if;
    ANS.FIRST := L.FIRST;
    ANS.LAST  := R.LAST;
    return ANS;
end f;
```

Figure 1: A Function to Help Solve the Maximal Plateau Problem

such applications are computing the number of peaks in a sequence of distinct terms (a term of the sequence is a "peak" if it is greater than both its predecessor and successor), computing the number of runs in a sequence, computing the maximum sum among the nonempty slices of a sequence, and computing the index of the first component of a vector whose value satisfies a particular property, such as being maximal, minimal, nonzero, or positive; if desired, the index of the last such component could be found instead.

LEMMA 4. *Let s and t be nonnull finite sequences of elements of a set E and let f be associative relative to $s \circ t$. Then $f_r(s \circ t) = f(f_r(s), f_r(t))$.*

PROOF. By induction on the length of s. First assume s has length 1, say s is $< x >$. Then

$$\begin{aligned} f_r(s \circ t) &= f_r(< x > \circ t) \\ &= f(x, f_r(t)) \\ &= f(f_r(s), f_r(t)). \end{aligned}$$

Now assume s has length greater than 1, say $s =< x > \circ u$, where u is nonnull and has length less than that of s. Then

$$\begin{aligned} f_r(s \circ t) &= f_r((< x > \circ u) \circ t) \\ &= f_r(< x > \circ (u \circ t)) \\ &= f(x, f_r(u \circ t)) \\ &= f(x, f(f_r(u), f_r(t))) \\ &= f(f_r(< x >), f(f_r(u), f_r(t))) \\ &= f(f_r(< x > \circ u), f_r(t)) \\ &= f(f_r(s), f_r(t)) \end{aligned}$$

where the third and fifth equalities follow from the definition of f_r, the fourth follows from the induction hypothesis, and the sixth follows from Lemma 2.□

3 CHARACTERIZATION THEOREMS

RELATIVE ASSOCIATIVITY CHARACTERIZATION THEOREM. *Let u be a nonnull finite sequence of elements of a set E, let f be a function from $E \times E$ to E, let S be the set of slices of u, and let R denote the smallest subset of $S \times E$ such that*

(a) *for each term x of u, R contains the pair $(< x >, x)$ and*

(b) *for each s in S, R contains each pair $(s, f(t', t''))$, where (s', t') and (s'', t'') are in R and s is s' concatenated with s''.*

Then the following statements are equivalent:

1. *f is associative relative to u.*
2. *R is a function from S to E.*
3. *R is a function from S to E that maps each s in S to $f_r(s)$.*

PROOF. First note that a smallest subset of $S \times E$ satisfying (a) and (b) exists, since $S \times E$ itself satisfies (a) and (b) and the intersection of all subsets of $S \times E$ that satisfy (a) and (b) satisfies (a) and (b) and obviously is included in each subset of $S \times E$ satisfying (a) and (b).

Since (3) clearly implies (2), it suffices to show that (1) implies (3) and (2) implies (1). To see that (1) implies (3), suppose f is associative relative to u. We show that for all s in S, any element of R whose first coordinate is s has second coordinate $f_r(s)$, by induction on the length of s. First assume s has length *1*, say s is $< x >$, and note that the only pair having first coordinate $< x >$ that needs to be in R for R to satisfy (a) and (b) is $(< x >, x)$. Since R is the smallest subset of $S \times E$ satisfying (a) and (b), it follows that the only element of R whose first coordinate is $< x >$ has second coordinate x, and thus the result holds, since $f_r(s)$ equals x. Now assume s has length greater than *1*. Since R is the smallest subset satisfying (a) and (b), the second coordinate of any pair in R whose first coordinate is s can be denoted as $f(t', t'')$, where (s', t') and (s'', t'') are in R and s is $s' \circ s''$. Each of s' and s'' is nonnull, since each is in S, so it follows from $s = s' \circ s''$ that each of s' and s'' has length less than that of s. By the induction hypothesis, $t' = f_r(s')$ and $t'' = f_r(s'')$ so

$$\begin{aligned} f(t', t'') &= f(f_r(s'), f_r(s'')) \\ &= f_r(s' \circ s'') \\ &= f_r(s) \end{aligned}$$

where the second equality follows from Lemma 4, using the fact that f is associative relative to u (and hence associative relative to $s' \circ s''$).

To see that (2) implies (1), suppose R is a function from S to E, say f_R. To see that f is associative with respect to u, let s_1, s_2, and s_3 be nonnull finite sequences such that $s_1 \circ s_2 \circ s_3$ is a slice of u. Then

$$\begin{aligned} f(f_r(s_1), f(f_r(s_2), f_r(s_3))) &= f(f_R(s_1), f(f_R(s_2), f_R(s_3))) \\ &= f(f_R(s_1), f_R(s_2 \circ s_3)) \\ &= f_R(s_1 \circ (s_2 \circ s_3)) \\ &= f_R((s_1 \circ s_2) \circ s_3) \\ &= f(f_R(s_1 \circ s_2), f_R(s_3)) \\ &= f(f(f_R(s_1), f_R(s_2)), f_R(s_3)) \\ &= f(f(f_r(s_1), f_r(s_2)), f_r(s_3)) \end{aligned}$$

where the second, third, fifth, and sixth equalities follow from (b) and the fourth follows from the associativity of string concatenation; this proof will thus be complete when the first and last equalities have been justified. This is accomplished by showing that, for each s in S, $f_R(s) = f_r(s)$. We proceed by induction on the length of s. First assume s has length 1, say s is $< x >$. Then

$$\begin{aligned} f_R(s) &= f_R(< x >) \\ &= x \\ &= f_r(< x >) \\ &= f_r(s) \end{aligned}$$

where the second equality follows from (a) and the third equality follows from the definition of f_r. Now assume s has length greater than 1, say $s = < x > ot$, where t is nonnull and has length less than that of s. Then

$$\begin{aligned} f_R(s) &= f_R(< x > ot) \\ &= f(f_R(< x >), f_R(t)) \\ &= f(x, f_R(t)) \\ &= f(x, f_r(t) \\ &= f_r(< x > ot) \\ &= f_r(s) \end{aligned}$$

where the first and last equality follow from the structure of s, the second follows from (b), the third follows from (a), the fourth follows from the induction hypothesis, and the fifth follows from the definition of f_r. □

When it is possible to prove that a function is in fact associative, rather than just associative relative to a sequence, the following simpler theorem can be used. For convenience, we denote the Relative Associativity Characterization Theorem and the Associativity Characterization Theorem by RACT and ACT,

respectively.

ASSOCIATIVITY CHARACTERIZATION THEOREM. *Let S denote the set of nonnull finite sequences of elements of a set E, let f be a function from $E \times E$ to E, and let R denote the smallest subset of $S \times E$ such that*

(a) *for each x in E, R contains the pair $(< x >, x)$ and*

(b) *for each s in S, R contains each pair $(s, f(t', t''))$, where (s', t') and (s'', t'') are in R and s is s' concatenated with s''.*

Then the following statements are equivalent:

1. *f is associative.*
2. *R is a function from S to E.*
3. *R is a function from S to E that maps each s in S to $f_r(s)$.*

PROOF. For each nonnull finite sequence u of elements of E, let R_u denote the relation defined in the statement of RACT and let S_u denote the set of nonnull finite slices of u. Note that the S above is the union of all the S_u.

That a smallest R exists follows from an argument similar to that in the first paragraph of the proof of RACT.

To complete this proof, it suffices to show that the following statements are equivalent:

1. f is associative.
2. f is associative relative to u, for each nonnull finite sequence u of E.
3. R_u is a function from S_u to E, for each nonnull finite sequence u of E.
4. R_u is a function from S_u to E that maps each s in S_u to $f_r(s)$, for each nonnull finite sequence u of E.
5. R is a function from S to E.
6. R is a function from S to E that maps each s in S to $f_r(s)$

.

By Lemma 3, (1) and (2) are equivalent. By RACT, (2), (3), and (4) are equivalent. Clearly (6) implies (5). Thus it suffices to show that (5) implies (3) and (4) implies (6), but both of these implications follow from the similarity of the definitions of R and R_u. [In this proof, note the necessity of being able to conclude that the values of $R_u(s)$ and $R_{u'}(s)$ agree, whenever f is associative relative to two nonnull finite sequences u and u' and the sequence s is in both sets S_u and $S_{u'}$.] □

4 APPLYING THE CHARACTERIZATION THEOREMS

Let f be a function defined by a subprogram written in a programming language. As pointed out in Section 1, a straightforward proof of the associativity of f based on the proof of correctness of the function subprogram and using the definition of associativity involves the consideration of p^4 cases, where p is the number of cases in the proof of correctness of a single application of f.

The next result shows that using the characterization theorems, instead of the definition of associativity, requires the consideration of no more than p cases.

PROPOSITION. *Let E be a set, let f be a function from $E \times E$ to E, where f is defined by a subprogram, and let p be the number of cases in a given proof of correctness of f.*
(a) If the result of reducing an arbitrary finite nonnull sequence of E by f has been specified and this specification is satisfied for each sequence consisting of a single term, then a proof of the associativity of f based on the given proof of correctness of f requires the consideration of at most p cases.
(b) Let u be a finite sequence of E. If the result of reducing an arbitrary nonnull slice of u by f has been specified and this specification is satisfied for each slice of u consisting of a single term, then a proof of the associativity of f relative to u based on the given proof of correctness of f requires the consideration of at most p cases.

PROOF. Proof of (a) using ACT. (A similar proof of (b) can be given using RACT.) Let R be the set containing the pairs $(s, \underline{s})$, where s is a nonnull sequence of E and $\underline{s}$ is the specified value of reducing s using f. By the hypothesis, R contains $(< x >, x)$ for each x in E. Let s' and s'' be arbitrary nonnull slices of an arbitrary sequence s of E such that s is the concatenation of s' and s''. By ACT, if $(s, f(\underline{s'}, \underline{s''}))$ is the unique element of R whose first coordinate is s, then f is associative. That is, to conclude that f is associative it suffices to show that $f(\underline{s'}, \underline{s''})$ is the unique specified reduction of s by f. Since only a single application of f is involved, at most p cases need to be considered, one for each of the cases in the given proof of correctness of f.□

Note that since ACT is a characterization of associativity, we are not required to prove a property stronger than the associativity of f in using the method in the above proof. Of course, the analogous statement for the use of RACT in proving relative associativity also holds. This method will now be illustrated for the function f in Example 2.

Let f be the function given in Figure 1 to help find the length of the longest plateau in a nondecreasing sequence of integers. The number of logical paths in the code for f is 24, since $24 = (4 + 2) \times 2 \times 2$. As will be shown in this

section, due to obvious properties of the code for f, only 6 cases, rather than 24, need to be considered in proving the correctness of a single application of f. It follows that a straightforward proof of the relative associativity of f using the definition of associativity and such a proof of correctness of f involves $6^4 = 1296$ cases. Since such a proof is both long and straightforward, such a proof is not presented in this paper.

The appendix illustrates how numerous ad-hoc arguments can reduce the length of a proof based on the definition of relative associativity. The present section demonstrates how using the method in the proof of the Proposition avoids the kind of time consuming and error-prone ad-hoc arguments used in the appendix.

Let u be any slice of a nondecreasing finite sequence of integers and let s be the natural corresponding sequence of records of type PLATEAU_TYPE; that is, the fields LEN, FIRST_LEN, and LAST_LEN of each term of s have the value 1, and the fields FIRST and LAST of the i^{th} term of s both have the same value as the i^{th} term of u.

A proof of the associativity of f relative to s using the method in the proof of the Proposition proceeds as follows. A specification of the effect of reducing an arbitrary nonnull slice of s using f is given in Example 2. Note that the reduction of s obviously has the specified value when s has a single term. Now let s' and s'' be arbitrary nonnull slices of s such that s is the concatenation of s' and s''. Let u' and u'' denote the subsequences of u corresponding to s' and s'', respectively; e.g. if s' is the sequence $< s_i, \ldots, s_j >$, then u' denotes the sequence $< u_i, \ldots, u_j >$. Let variables L and R be assigned the specified reduction of s' and s'', respectively, so that:

- L.LEN is the length of the longest plateau in u'.
- L.FIRST_LEN is the length of the longest plateau in u' that starts at the first component of u'.
- L.LAST_LEN is the length of the longest plateau in u' that ends at the last component of u'.
- L.FIRST is the value of the first component of u'.
- L.LAST is the value of the last component of u'.

and

- R.LEN is the length of the longest plateau in u''.
- R.FIRST_LEN is the length of the longest plateau in u'' that starts at the first component of u''.
- R.LAST_LEN is the length of the longest plateau in u'' that ends at the last component of u''.

- R.FIRST is the value of the first component of u''.
- R.LAST is the value of the last component of u''.

Clearly the specified value of the reduction of any nondecreasing finite sequence by f is unique, due to the nature of the specification of f. Thus, due to RACT, we can conclude that f is associative if we show that f(L, R) is the specified reduction of s by f. We present a proof of this fact having the same amount of detail as the proof in the appendix.

```
if L.LAST = R.FIRST
   Then L.LAST_LEN + R.FIRST_LEN is the length of a plateau so
   ANS.LEN = max (max (L.LEN, R.LEN), L.LAST_LEN + R.FIRST_LEN)
   is the length of the longest plateau in u.
if L.LAST ≠ R.FIRST
   Then no plateau extends from the end of u' to the beginning of
   u'' so ANS.LEN = max (L.LEN, R.LEN) is the length of the
   longest plateau in u.
if L.FIRST = R.FIRST
   Then a plateau extends from the beginning of u' into u'' so
   ANS.FIRST_LEN = L.LEN + R.FIRST_LEN is the length of the
   longest plateau starting at the first component of u.
if L.FIRST ≠ R.FIRST
   Then no plateau extends from the beginning of u' into u'' so
   ANS.FIRST_LEN = L.LEN  is the length of the longest plateau
   starting at the first component of u.
if L.LAST = R.LAST
   Then a plateau extends from the end of u' to the end of u'' so
   ANS.LAST = L.LAST_LEN + R.LEN is the length of the longest
   plateau ending at the last component of u.
if L.LAST ≠ R.LAST
   Then no plateau extends from the end of u' to the end of u'' so
   ANS.LAST = R.LAST_LEN is the length of the longest plateau ending
   at the last component of u.
```

Finally, since u is u' concatenated with u'', ANS.FIRST and ANS.LAST are assigned the specified values. □

5 COMPOSITES OF ASSOCIATIVE FUNCTIONS

The function f in Example 2 is defined in terms of simpler functions. A natural question arises when such functions must be shown to be associative: Can an associative function, or a function that is associative relative to a sequence, be characterized as the composite of certain well-behaved functions? The most natural candidate for "well-behaved functions" are the associative functions themselves and such a characterization would be quite useful, since many functions that need to be shown associative, or associative relative to a sequence, are easily shown to be composites of associative functions.

On the one hand, an associative function f can always be written as the composite of associative functions, namely

$$f(x,y) = f(p_1(x,y),p_2(x,y))$$

where p_1 and p_2 are the projection functions $p_1(x,y) = x$ and $p_2(x,y) = y$, which are easily seen to be associative. Does the converse hold; i.e. is the composite of associative functions necessarily associative, or at least associative relative to a nontrivial sequence? (We view as trivial any sequence consisting entirely of zeroes, as well as any sequence having less than three terms, for which relative associativity is of no importance.) The answer is "no":

EXAMPLE 4. *There are associative functions f_1, f_2, and f_3 such that their composite f, defined by*

$$f(x,y) = f_1(f_2(x,y),f_3(x,y))$$

fails to be associative relative to any nontrivial sequence. For let $f_1(x,y) = x+y$, let f_2 and f_3 be the projection function p_1, defined above, and let f be defined by

$$f(x,y) = f_1(f_2(x,y),f_3(x,y))$$

Then $f(x,y) = x + x = 2x$, which is not associative since $f(x,f(y,z)) = 2x$ whereas $f(f(x,y),z) = 4x$. It is easy to see that f also fails to be associative relative to any nontrivial sequence.□

APPENDIX

Let f be the function given in Figure 1 to help find the length of the longest plateau in a sequence. Let u be any nondecreasing sequence of integers and let s be the natural corresponding sequence of records of type PLATEAU_TYPE; that is, the fields LEN, FIRST_LEN, and LAST_LEN of each term of s have the value 1, and the fields FIRST and LAST of the i^{th} term of s both have the same value as the i^{th} term of u. In Section 4.1 f is proven to be associative relative to s using the method in the proof of the Proposition of Section 4. In this section f is shown to be associative relative to s using the definition of associativity and ad-hoc arguments based on special properties of f.

LEMMA. *Let* $t =< t_i, \ldots, t_j >$ *be any slice of* s, *let* T *be* $f_r(t)$, *and let* $u_t =< u_i, \ldots, u_j >$. *Then* T.LEN *is the length of the longest plateau in* u_t, T.FIRST_LEN *is the length of the longest plateau in* u_t *that starts at the first component of* u_t, T.LAST_LEN *is the length of the longest plateau in* u_t *that ends at the last component of* u_t, T.FIRST *is the value of the first component of* u_t, *and* T.LAST *is the value of the last component of* u_t.

PROOF. By induction on the length of t. The Lemma clearly holds for slices of length 1, so assume it holds for slices of length n and let t be a slice of length $n+1$, say $t =< x > \circ v$. Since $f_r(t) = f(x, f_r(v))$, T results from a call on f for which L.LEN, L.FIRST_LEN, and LAST_LEN have value 1 and L.FIRST, L.LAST have equal values, a value that is no greater than R.FIRST, and by the induction hypothesis, R satisfies the above conditions. It is then straightforward to check that the value ANS returned by this call satisfies the above conditions.□

PROPOSITION. *The function* f *is associative relative to the sequence* s.

PROOF. To shorten the proof, we replace **if** statements by conditional expressions. For example, the statement:

```
if L.FIRST = R.FIRST then
  ANS.FIRST_LEN := L.LEN + R.FIRST_LEN;
else
  ANS.FIRST_LEN := L.FIRST_LEN;
end if;
```

is replaced by

```
ANS.FIRST_LEN := [if L.FIRST = R.FIRST then L.LEN + R.FIRST_LEN
                                       else L.FIRST_LEN];
```

Let $s_1 \circ s_2 \circ s_3$ be a slice of s, let u_1, u_2, and u_3 be the slices of u corresponding to s_1, s_2, and s_3, and let X, Y, and Z denote $f_r(s_1)$, $f_r(s_2)$, and $f_r(s_3)$, respectively.

```
TO SHOW:  f(X, f(Y,Z)).x = f(f(X,Y), Z).x
for x = LEN, FIRST_LEN, LAST_LEN, FIRST, and LAST.

f(X, f(Y,Z)).LEN
  = [if X.LAST = f(Y,Z).FIRST then
       max (X.LEN, f(Y,Z).LEN, X.LAST_LEN + f(Y,Z).FIRST_LEN)
     else
       max(X.LEN, f(Y,Z).LEN)]
  = [if X.LAST = Y.FIRST then
       max (X.LEN,
            [if Y.LAST = Z.FIRST then
                 max (Y.LEN, Z.LEN, Y.LAST_LEN + Z.FIRST_LEN)
             else max (Y.LEN, Z.LEN)],
            X.LAST_LEN + [if Y.FIRST = Z.FIRST then
                             Y.LEN + Z.FIRST_LEN
                          else Y.FIRST_LEN])
     else
       max (X.LEN,
            [if Y.LAST = Z.FIRST then
                 max (Y.LEN, Z.LEN, Y.LAST_LEN + Z.FIRST_LEN)
             else max (Y.LEN, Z.LEN)])
    ]
  = [if Y.LAST = Z.FIRST then
       max ([if X.LAST = Y.FIRST then
                 max (X.LEN, Y.LEN, X.LAST_LEN + Y.FIRST_LEN)
             else max (X.LEN, Y.LEN)],
            Z.LEN,
            [if X.LAST = Y.LAST then X.LAST_LEN + Y.LEN else Y.LAST_LEN]
             + Z.FIRST_LEN)
     else
       max ([if X.LAST = Y.FIRST then
                 max (X.LEN, Y.LEN, X.LAST_LEN + Y.FIRST_LEN)
             else max (X.LEN, Y.LEN)],
            Z.LEN)
    ]

  Proof of the last equality:
  if Y.LAST = Z.FIRST
     if X.LAST = Y.FIRST
        if Y.FIRST = Z.FIRST  Then clearly X.LAST = Y.LAST.
                      The first expression is equal to
                      max (X.LEN, Y.LEN, Z.LEN, Y.LAST_LEN + Z.FIRST_LEN,
                           X.LAST_LEN + Y.LEN + Z.FIRST_LEN)
                      and the second expression is equal to
                      max (X.LEN, Y.LEN, X.LAST_LEN + Y.FIRST_LEN,
                           Z.LEN, X.LAST_LEN + Y.LEN + Z.FIRST_LEN)
                      Now since Y.FIRST = Y.LAST, all of u2 is a plateau so
```

```
                    X.LAST_LEN + Y.LEN + Z.FIRST_LEN >=
                    Y.LAST_LEN + Z.FIRST_LEN and
                    X.LAST_LEN + Y.LEN + Z.FIRST_LEN >=
                    X.LAST_LEN + Y.FIRST_LEN so the two
                    expressions are equal.
      if Y.FIRST # Z.FIRST  Then clearly X.LAST # Y.LAST.
                    The first expression is equal to
                    max (X.LEN, Y.LEN, Z.LEN, Y.LAST_LEN + Z.FIRST_LEN,
                         X.LAST_LEN + Y.FIRST_LEN)
                    and the second expression is equal to
                    max (X.LEN, Y.LEN, X.LAST_LEN + Y.FIRST_LEN, Z.LEN,
                         Y.LAST_LEN + Z.FIRST_LEN)
                    so the two expressions are equal.
   if X.LAST # Y.FIRST  Then X.LAST # Y.LAST since the sequence is nondecreasing.
                  Thus both expressions equal
                  max (X.LEN, Y.LEN, Z.LEN, Y.LAST_LEN + Z.FIRST_LEN)
if Y.LAST # Z.FIRST  Then Y.FIRST # Z.FIRST since the sequence is nondecreasing.
   if X.LAST = Y.FIRST  The first expression is equal to
                  max (X.LEN, Y.LEN, Z.LEN, X.LAST_LEN + Y.FIRST_LEN)
                  and the second expression is equal to
                  max (X.LEN, Y.LEN, X.LAST_LEN + Y.FIRST_LEN, Z.LEN)
                  so the two expressions are equal.
   if X.LAST # Y.FIRST  Both expressions equal max (X.LEN, Y.LEN, Z.LEN).

= [if Y.LAST = Z.FIRST then
     max ([if X.LAST = Y.FIRST then
                max (X.LEN, Y.LEN, X.LAST_LEN + Y.FIRST_LEN)
           else max (X.LEN, Y.LEN)],
          Z.LEN,
          f(X,Y).LAST_LEN + Z.FIRST_LEN)
   else
     max ([if X.LAST = Y.FIRST then
                max (X.LEN, Y.LEN, X.LAST_LEN + Y.FIRST_LEN)
           else max (X.LEN, Y.LEN)],
          Z.LEN)
= [if Y.LAST = Z.FIRST then
     max (f(X,Y).LEN, Z.LEN, f(X,Y).LAST_LEN + Z.FIRST_LEN)
   else
     max (f(X,Y).LEN,
          Z.LEN)
= [if f(X,Y).LAST = Z.FIRST then
     max (f(X,Y).LEN, Z.LEN, f(X,Y).LAST_LEN + Z.FIRST_LEN)
   else
     max (f(X,Y).LEN,
          Z.LEN)]
= f(f(X,Y), Z).LEN
```

```
f(X, f(Y,Z)).FIRST_LEN
  = [if X.FIRST = f(Y,Z).FIRST then X.LEN + f(Y,Z).FIRST_LEN else X.FIRST_LEN]
  = [if X.FIRST = Y.FIRST then X.LEN +
       [if Y.FIRST = Z.FIRST then Y.LEN + Z.FIRST_LEN else Y.FIRST_LEN]
     else X.FIRST_LEN]
  = [if X.FIRST = Z.FIRST then
       [if X.LAST = Y.FIRST then
            max (X.LEN, Y.LEN, X.LAST_LEN + Y.FIRST_LEN)
        else max (X.LEN, Y.LEN)] + Z.FIRST_LEN
     else
       [if X.FIRST = Y.FIRST then X.LEN + Y.FIRST_LEN else X.FIRST_LEN]]

  Proof of the last equality:
  if X.FIRST = Y.FIRST  Then X.LAST = Y.FIRST, since the sequence is nondecreasing.
     if Y.FIRST = Z.FIRST  Then X.FIRST = Z.FIRST.
                  The first expression is equal to
                  X.LEN + Y.LEN + Z.FIRST_LEN
                  and the second expression is equal to
                  max (X.LEN, Y.LEN, X.LAST_LEN + Y.FIRST_LEN) + Z.FIRST_LEN.
                  Note that, since X.FIRST = Z.FIRST, and since the
                  sequence is nondecreasing, all terms from
                  X.FIRST to Z.FIRST have the same value; thus
                  X.LAST_LEN + Y.FIRST_LEN = X.LEN + Y.LEN so
                  X.LAST_LEN + Y.FIRST_LEN = max (X.LEN, Y.LEN,
                                                  X.LAST_LEN + Y.FIRST_LEN) so
                  the second expression is also equal to
                  X.LEN + Y.LEN + Z.FIRST_LEN.
     if Y.FIRST # Z.FIRST  Then X.FIRST # Z.FIRST.
                  Both expressions equal X.LEN + Y.FIRST_LEN.
  if X.FIRST # Y.FIRST  Then X.FIRST # Z.FIRST, since the sequence is nondecreasing.
               Both expressions equal X.FIRST_LEN.

  = [if f(X,Y).FIRST = Z.FIRST then f(X,Y).LEN + Z.FIRST_LEN else f(X,Y).FIRST_LEN]
  = f(f(X,Y), Z).FIRST_LEN


f(X, f(Y,Z)).LAST_LEN
  = [if X.LAST = f(Y,Z).LAST then X.LAST_LEN + f(Y,Z).LEN else f(Y,Z).LAST_LEN]
  = [if X.LAST = Z.LAST then X.LAST_LEN + f(Y,Z).LEN else f(Y,Z).LAST_LEN]
  = [if X.LAST = Z.LAST then
       X.LAST_LEN + [if Y.LAST = Z.FIRST then
                        max (Y.LEN, Z.LEN, Y.LAST_LEN + Z.FIRST_LEN)
                     else max (Y.LEN, Z.LEN)]
     else [if Y.LAST = Z.LAST then Y.LAST_LEN + Z.LEN else Z.LAST_LEN]]

  = [if Y.LAST = Z.LAST then
       [if X.LAST = Y.LAST then X.LAST_LEN + Y.LEN else Y.LAST_LEN] + Z.LEN
```

```
   else Z.LAST_LEN]

  Proof of the last equality:
  if X.LAST = Z.LAST  Then X.LAST = Y.LAST = Z.FIRST = Z.LAST, since the
                sequence is nondecreasing.
                The first expression equals
                X.LAST_LEN + max (Y.LEN, Z.LEN, Y.LAST_LEN + Z.FIRST_LEN)
                The second expression equals
                X.LAST_LEN + Y.LEN + Z.LEN
                Note that the first expression can be rewritten as
                X.LAST_LEN + Y.LAST_LEN + Z.FIRST_LEN since
                Y.LAST_LEN + Z.FIRST_LEN = Y.LEN + Z.LEN
                so the two expressions are equal.
  if X.LAST # Z.LAST
     if Y.LAST = Z.FIRST
        if Y.LAST = Z.LAST  Then X.LAST # Y.LAST.
                        Both expressions equal Y.LAST_LEN + Z.LEN.
        if Y.LAST # Z.LAST  Both expressions equal Z.LAST_LEN
     if Y.LAST # Z.FIRST  Then Y.LAST # Z.LAST, since the sequence is nondecreasing.
                     Both expressions equal Z.LAST_LEN.

  = [if f(X,Y).LAST = Z.LAST then
       [if X.LAST = Y.LAST then X.LAST_LEN + Y.LEN else Y.LAST_LEN] + Z.LEN
     else Z.LAST_LEN]
  = [if f(X,Y).LAST = Z.LAST then f(X,Y).LAST_LEN + Z.LEN else Z.LAST_LEN]
  = f(f(X,Y), Z).LAST_LEN

f(X, f(Y,Z)).FIRST
  = X.FIRST
  = f(X,Y).FIRST
  = f(f(X,Y), Z).FIRST

f(X, f(Y,Z)).LAST
  = f(Y,Z).LAST
  = Z.LAST
  = f(f(X,Y), Z).LAST
```

7 REFERENCES

[Cha90] Charlesworth, A. The nondeterministic divide. Tech. Rep. IPC-TR-90-005, Inst. for Parallel Computation, U. of Virginia, Nov. 1990.

[Gri81] Gries, D. *The Science of Programming.* Springer-Verlag, New York, 1981.

[HS86] Hillis, W. D. and Steele, G. L., Jr. Data parallel algorithms. *Comm. ACM*, 29, 12, (Dec. 1986), 1170-1183.

[Int89] Intel Corporation, *iPSC/2 Programmer's Reference Manual.* Beaverton, Or., Oct. 1989.

[Ive62] Iverson, K. E., *A Programming Language.* John Wiley and Sons, New York, 1962.

[Sab88] Sabot, G. *The Paralation Model: Architecture-Independent Parallel Programming.* MIT Press, Cambridge, 1988.

INSTITUTE FOR PARALLEL COMPUTATION
RECENT TECHNICAL REPORTS

87-001 *Basic Database Concepts in ADAMS (Advanced Data Manipulation System): Language Interface for Process Service.* J.L. Pfaltz, S.H. Son, J.C. French, P.K. Baron, D.J. Kirks, and R. Orlandic, November 30, 1987.

88-001 *Compact O-Complete Trees: A New Method for Searching Large Files.* R. Orlandic and J.L. Pfaltz, January 26, 1988.

88-002 *Reliability Mechanisms for ADAMS.* S.H. Son and J.L. Pfaltz, March 20, 1988.

88-003 *Scoping Persistent Name Spaces in ADAMS.* J.L. Pfaltz, J.C. French and J.L. Whitlatch, June 28, 1988.

88-004 *Implementing Set Operators Over Class Hierarchies.* J.L. Pfaltz, August 5, 1988.

88-005 *Implementation of an ADAMS Prototype: The ADAMS Preprocessor (AP).* C. Klumpp and J.L. Pfaltz, August 9, 1988.

88-006 *The 1988 Parallel Sorting Bibliography.* D. Richards, August 25, 1988.

88-007 *A Spectrum of Options for Parallel Simulation.* P.F. Reynolds, September 9, 1988.

88-008 *A Neural Network Implementation of a Correspondence Processing Algorithm.* A. Barker, D.E. Brown and W. Martin, October 9, 1988.

88-009 *A Prototyping Environment for Distributed Database Systems: Functional Description.* S.H. Son, J. Ratner and C-H. Chang, October 9, 1988.

88-010 *A Global Time Reference for Hypercube Multicomputers.* J.C. French, October 10, 1988.

88-011 *A Procedure for Generating Source Weights in Group Consensus Problems.* D.E. Brown and M. Mostaghimi, December 12, 1988.

89-001 *A Bibliography of Heuristic Search.* B.S. Stewart, January 30, 1989.

89-002 *The ADAMS Database Language.* J.L. Pfaltz, J.C. French, A.S. Grimshaw, S.H. Son, P.K. Baron, S. Janet, A. Kim, C. Klumpp, Y. Lin, L. Loyd, February 28, 1989.

89-003 *A Parallel Heuristic for Quadratic Assignment Problems.* C.L. Huntley and D.E. Brown, March 16, 1989.

89-004 *A Method for the Evaluation of Correlation Algorithms.* A.R. Spillane, C.L. Pittard, Jr.

and D.E. Brown, April 1, 1989.

89-005 *A Justification for Applying the Principle of Minimum Relative Entropy to Information Integration Problems*. D.E. Brown, April 18, 1989.

89-006 *Design, Analysis and Applications of Compact 0-Complete Trees*. R. Orlandic, May 22, 1989.

89-007 *Performance Evaluation of Multiversion Database Systems*. S.H. Son and N. Haghighi, July 25, 1989.

89-008 *The ADAMS Storage Management System*. S.A. Janet, Jr., August 10, 1989.

89-009 *The ADAMS Preprocessor*. P.K. Baron, December 4, 1989.

89-010 *Implementation of the ADAMS Database System*. J.L. Pfaltz, J.C. French, A.S. Grimshaw, S.H. Son, P.K. Baron, S. Janet, Y. Lin, L. Loyd, R. McElrath, December 11, 1989.

89-011 *Comparative Analyses of Parallel Simulation Protocols*. P.F. Reynolds, Jr., C. Weight and J.R. Fidler, December 6, 1989.

89-012 *SPECTRUM: A Parallel Simulation Testbed* P.F. Reynolds, Jr. and P.M. Dickens, **March 12, 1989.**

90-001 *Uncertainty Management with Imprecise Knowledge with Application to Design*. D.E. Brown and W.J. Markert, January 4, 1990.

90-002 *ASSET: A Simulation Test Bed for Evaluating Data Association Algorithms*. D.E. Brown, C.L. Pittard, and A.R. Spillane, January 22, 1990.

90-003 *SRADS With Local Rollback*. P.M. Dickens and P.F. Reynolds, Jr., January 22, 1990.

90-004 *Genetic Algorithms for Feature Selection for Counterpropogation Networks*. F.Z. Brill, D.E. Brown and W.N. Martin, April 9, 1990.

90-005 *The Nondeterministic Divide*. Arthur Charlesworth, November, 1990.

90-006 *Parallel Genetic Algorithms with Local Search*. Christopher L. Huntley and Donald E. Brown, September 3, 1990.

90-007 *A Characterization of Associativity*. Arthur Charlesworth, November 1990.