MATHEMATICS OF COMPUTATION, VOLUME 32, NUMBER 142
APRIL 1978, PAGES 593–605

Deciding Associativity for Partial Multiplication Tables of Order 3*

By Paul W. Bunting, Jan van Leeuwen and Dov Tamari

Abstract. The associativity problem for the class of finite multiplication tables is known to be undecidable, even for quite narrow infinite subclasses of tables. We present criteria which can be used to decide associativity in many cases, although any effective method based on such criteria must eventually fail on a table of some size (as otherwise decidability for the general class would follow). By means of an extensive computer search we have been able to use the criteria successfully to solve the associativity problem for all tables of order up to 3. We find 24,733 associative tables and 237,411 nonassociative tables, and present some further statistics about how "deep" we had to search to establish the nonassociativity of a table. We also prove that there are tables of order 3 for which no "one-mountain" theorem holds (which was known previously only for order 6 examples). Our methods make use of efficient data-representations and techniques of heuristic and adaptive programming.

1. Introduction. A finite partial multiplication table (or simply: a table) is a set $V = \{v_1, \ldots, v_n\}$ with an $n \times n$ array $A = (a_{ij})$ such that

$$a_{ij} = \begin{cases} v_k, & \text{if } v_i \cdot v_j = v_k, \\ *, & \text{if } v_i \cdot v_j \text{ is undefined (a hole).} \end{cases}$$

The associativity problem is the question to determine for any given table $\langle V, A\rangle$ whether or not it can be embedded in a semigroup table.

Tamari proved that the associativity problem for tables is algorithmically undecidable, by reducing it first to the word-problem for groups [17], [18] and later to the word-problem for semigroups [19], [20]. In this paper we shall present several criteria which can be used to decide associativity in many cases, although clearly any general effective method based on such criteria must eventually fail on a table of some size (as otherwise decidability for the general class would follow).

Earlier Tamari classified all $3^4 = 81$ tables of order 2 by hand. As there are as many as $4^9 = 262{,}144$ tables of order 3 (admittedly containing many isomorphic tables), it is no longer feasible to proceed to higher-order tables by mere hand-calculation. The main result of this paper is that by means of an extensive computer search we have been able to use our set of criteria successfully to solve the associativity problem for all tables of order up to 3.

We report how our criteria are used to have the computer drastically reduce the number of tables requiring closer examination. Typically, we apply a number of in-

Received March 31, 1977.

AMS (MOS) subject classifications (1970). Primary 68A15, 02F47, 05–04.

*This work is based on the first author's M. Sc. thesis, completed at the Department of Computer Science, SUNY at Buffalo, Amherst, New York 14226.

expensive tests first to obtain an initial separation of associative, nonassociative, and "still undecided" tables. We do not use isomorphism-rejection until a rather late stage in the process. In the end we must solve 16 remaining cases by hand. We find 24,733 associative tables and 237,411 nonassociative tables, and present some further statistics about how "deep" we had to search to establish the nonassociativity of tables. We also prove that Tamari's "one-mountain" theorem [20] does not hold for general tables, by means of order 3 examples. (It was demonstrated previously only by means of order 6 examples.)

There would be as many as $5^{16} = 152{,}587{,}890{,}625$ cases to consider if we were to extend our classification to tables of order 4. Even with isomorphism-rejection under a plausible group of symmetries in advance, we expect that our current criteria and heuristics will require 4 months of computing time (assuming that our criteria are indeed sufficient for tables of order up to 4). The associativity problem for tables of order 5 seems to be well beyond our computing capacity.

It is conceivable that more intricate rejection-criteria can reduce the computing time for tables of order 4 to within affordable limits. We hope that our present results for tables of order 3 provide some new insights into the intricate structure of multiplication-tables, in view of the continuing efforts to analyze and catalog finite semigroups and related structures by computer. We mention the studies of Forsythe [6], Tetsuya et al. [21], Plemmons [13], [14], and Jürgensen et al. [7]–[9]. Recently Jürgensen and Wick [9] completed the classification of all semigroup tables of order up to 7.

Our computer programs make use of efficient data-representations and techniques of heuristic and adaptive programming from artificial intelligence. We benefitted from the studies of Weiss [22] and Baker, Dewdney and Szilard [1] and Bitner and Reingold [2] in deciding some details in implementing our criteria.

2. Preliminaries. Let $V = \{v_1, \dots, v_n\}$ be an alphabet (or set of generators), and let V^* be the set of all words (strings) of finite length over V. Each string can be considered as a product in the generators, provided we fix a particular order of evaluation for the expression. Informally, a table T is called associative if for all strings the evaluation strategies which yield a single letter value all produce, in fact, an identical such value. T is called nonassociative if there is a string for which at least two different strategies yield different single letter values.

2.1. *Notations and Theoretical Results.* Let T be an arbitrary table over V. Let $X, Y, \alpha, \beta, \dots$ be strings in V^*. We say that X contracts to Y, notation $X \xrightarrow[c]{} Y$, if we can write $X = \alpha v_i v_j \beta$ and $Y = \alpha v_k \beta$ with $v_i v_j = v_k$. We say that X expands to Y, notation $X \xrightarrow[e]{} Y$, if we can write $X = \alpha v_k \beta$ and $Y = \alpha v_i v_j \beta$ with $v_i v_j = v_k$. We shall write $X \xrightarrow[c,e]{} Y$ if and only if $X \xrightarrow[c]{} Y$ or $X \xrightarrow[e]{} Y$ (where it should be noted that the two relations cannot be valid simultaneously). Let $\xrightarrow[c,e]{*}$ denote the reflexive and transitive closure of $\xrightarrow[c,e]{}$. We may occasionally want to add a subscript T to indicate the specific table to which the relations refer.

Let $X \equiv_T Y$ if and only if $X \xrightarrow[c,e]{*} Y$. One can verify that $\equiv_T$ is an equivalence relation. The associativity of a table may now be formulated as follows.

Definition. Table T is associative if and only if $v_i \not\equiv_T v_j$ for all $i \neq j$.

A pair v_i, v_j $(i \neq j)$ with $v_i \equiv_T v_j$ will be called a "contradiction" for table T.

In order to determine the (non)associativity of a table it would be of help to know how contradictions can occur. Unfortunately, little is known, except for Tamari's "one-mountain" theorem for symmetric tables ([20], also Newman [11]). A table is called "symmetric" if it is the table of a partial inverse property loop (or symmetrical monoid, de Carvalho and Tamari [5], Tamari [18], [20]).

THEOREM 2.a. *If T is symmetric and $v_i \equiv_T v_j$ $(i \neq j)$, then there is an $X \in V^*$ such that $v_i \xrightarrow[e]{*} X \xrightarrow[c]{*} v_j$.*

Proof. Let $v_i \xrightarrow[c,e]{*} v_j$, and consider a chain of contractions and expansions from v_i to v_j. Suppose that somewhere in the chain it happens that a contraction precedes an expansion. Consider the last contraction in the chain with this property

$$v_i \xrightarrow[c,e]{} \cdots \xrightarrow[c,e]{} X_1 \xrightarrow[c]{} X_2 \xrightarrow[e]{} \cdots \xrightarrow[e]{} X_t \xrightarrow[c]{*} v_j \quad (\text{some } t > 2).$$

Let $X_1 = \alpha v_k v_l \beta$ and $X_2 = \alpha v_m \beta$ with $v_k \cdot v_l = v_m$. Since T is symmetric one must have $v_l = v_k^{-1} v_m$, and one can change the beginning of the segment from X_1 to X_t to

$$\cdots (X_1 =) \alpha v_k v_l \beta \xrightarrow[e]{} \alpha v_k v_k^{-1} v_m \beta \longrightarrow \cdots$$

where we observe that the resulting string properly "contains" X_2. It follows that we can proceed with the expansions which were applied to $X_2, \ldots, X_{t-1}$ originally, and obtain X_t back after contracting $v_k v_k^{-1}$ to 1 (the identity of T) and one more contraction to absorb the 1.

By repeating this construction one can move all contractions to the end of the chain. □

The one-mountain theorem asserts that under special circumstances one may find all contradictions in a table by restricting attention to chains in which all expansions occur before any contractions. A one-mountain theorem was found recently also in the entirely different context of recursion-structure simplification (Strong, Maggiolo-Schettini and Rosen [16]).

Unfortunately, the one-mountain theorem (2.a) is not true in general. Tamari [20] gave a counterexample in a table of order 6, and we shall find counterexamples even in tables of order 3. Even when the one-mountain theorem holds it will not be of great help in "solving" the associativity problem, as Tamari [17], [18] showed that the problem is even undecidable for the class of symmetric tables (by a reduction to the word-problem for groups).

One may want to classify nonassociative tables according to the minimal chain-length needed to find a contradiction: the smallest k such that $v_i \xrightarrow[c,e]{k} v_j$ for some $i \neq j$. Denote this value for table T by k_T.

Definition. $L(n) = \max\{k_T | T \text{ a (nonassociative) table of order } n\}$.

L is seen to be a nondecreasing, nonrecursive total function. Knowing L (as an oracular input) would enable us to bound the search for contradictions in a table.

Open Problem. Prove or disprove that L is a strictly increasing function.

Once a table is known to be associative (nonassociative) one can immediately derive the conclusion for some other tables. S is a subtable of T if and only if S is obtained from T by replacing some defined products by *'s (holes). S is a supertable of T if and only if T is a subtable of S.

PROPOSITION 2.b. *If T is associative, then so are all subtables of T.*

PROPOSITION 2.c. *If T is nonassociative, then so are all supertables of T.*

Testing for associativity is sometimes easy. A full table (without *'s) is associative if and only if

$$v_i(v_j v_k) = (v_i v_j) v_k$$

for all $1 \leqslant i, j, k \leqslant n$ (see Clifford and Preston [4] or Bruck [3] for additional details). Proposition 2.b immediately shows that subtables of known semigroup tables are associative.

If a table contains *'s (holes) to begin with, then one must inspect all expressions in the generators (of size 3, 4, ...) and all evaluation-strategies for each string in order to find a contradiction (if there is one) or to obtain the basis of a finite proof that no contradictions can occur. Heuristically, the size of expressions for consideration can usually be kept rather low, as long as there are only a few *'s in the table. We shall find more evidence for it in Section 3.

A useful technique for speeding up the search for possible contradictions is table-completion.

Definition. S is a completion of T if and only if there is a sequence of tables $T_0, T_1, \ldots, T_p$ such that

(a) $T_0 = T$,

(b) $T_p = S$,

(c) for $0 \leqslant l < p$, T_l is a subtable of T_{l+1},

(d) for each l $(0 \leqslant l < p)$, if $v_i \cdot v_j = *$ in T_l and $v_i \cdot v_j = v_k$ in T_{l+1} then $v_i \cdot v_j \equiv_{T_l} v_k$.

A table-completion is obtained by "filling up" more and more holes, in a manner which is consistent with the current products.

THEOREM 2.d. *T is (non)associative if and only if each completion S of T is (non)associative.*

Proof. We shall prove by induction that for all X, Y: $X \equiv_T Y$ if and only if $X \equiv_S Y$.

Let S be obtained from T by a chain of consecutive completions $T = T_0, T_1, \ldots, T_p = S$. Assume (as we may) that each completion is obtained by filling in one * at a time. A contraction or expansion which uses a non-* entry from T_l will be called a T_l-step. We claim that for $0 \leqslant l \leqslant p$: $X \equiv_T Y$ if and only if $X \equiv_{T_l} Y$.

The claim is trivial for $l = 0$. For $l > 0$ we shall prove the claim by showing that for $0 \leqslant l < p$: $X \equiv_{T_l} Y$ if and only if $X \equiv_{T_{l+1}} Y$.

If $X \equiv_{T_l} Y$, then certainly $X \equiv_{T_{l+1}} Y$. Let T_{l+1} be obtained from T_l by writing $v_i v_j = v_k$ for $v_i v_j = *$ in T_l, with $v_i v_j \equiv_{T_l} v_k$. Let $X \equiv_{T_{l+1}} Y$ and consider a chain of T_{l+1}-steps

$$X = X_0 \underset{c,e}{\rightarrow} \cdots \underset{c,e}{\rightarrow} X_m = Y.$$

Consider each step which is not a T_l-step. If the step is a contraction, then it must be of the form

$$\alpha v_i v_j \beta \underset{c}{\to} \alpha v_k \beta$$

with a reference to the new product. As $v_i v_j \equiv v_k$ in T_l, we can replace this step by an explicit chain of T_l-steps

$$\alpha v_i v_j \beta \underset{c,e}{\to} \cdots \underset{c,e}{\to} \alpha v_k \beta.$$

One can make a dual replacement if the non-T_l step is an expansion. In either case we can rewrite the chain entirely in T_l-steps. It follows that $X \equiv_{T_l} Y$. □

In later associativity tests we shall indeed always attempt to complete a table as far as we can (within reasonable search limits). If we find conflicting products in doing so, then we obviously have a nonassociative table. As long as no conflicting products are found, we may hope that each table-completion gives a valuable shortcut in the many evaluation-strategies we must try in order to eventually arrive at a decision.

2.2. *Computer Representation of Tables.* To facilitate the processing of tables of order n by computer we need a simple, numeric representation of the "values" from V which appear in the boxes of the $n \times n$ array: we shall code * as 0, v_1 as 1, ... , v_n as n.

Example:

*	v_1	v_2
v_3	*	v_1
*	v_3	v_2

is represented as

0	1	2
3	0	1
0	3	2

Let $T(i, j)$ denote the numeric entry for $v_i v_j$ in table T.

For the purpose of storing tables in (external) files we need a simple scheme to code entire tables into numbers. We shall represent each table T of order n as an $(n + 1)$-ary number $T(1, 1)T(1, 2) \cdots T(n, n)$ or

$$\text{code}(T) = \sum_{i=1}^{n} \sum_{j=1}^{n} T(n + 1 - i, n + 1 - j) \cdot (n + 1)^{n(i-1)+(j-1)}.$$

Straightforward routines were written to convert tables into codes and conversely, to store codes into a file and to read codes from a file. A routine was written to print tables in readable form on a teletype, so as to facilitate direct inspection and tests by hand.

All routines were written in PASCAL and run on the CDC 6400 computer at the Computation Center of SUNY at Buffalo. A listing of the programs is available from the authors.

3. Deciding (Non)Associativity for Tables of Order 3. We shall proceed to explain the techniques by means of which we were able to decide associativity for all 262,144 tables of order 3.

Our general strategy for solving the associativity problem is as follows. We begin with a few promising, expedient criteria which eliminate (i.e., decide) a large number of tables right away at a low cost per table. Based on a study of some of the remaining tables we choose a new criterion which is likely to eliminate many more tables with little extra effort, and repeat this heuristic elimination-process until we get stuck.

We consider the order in which we applied the various criteria at least as interesting as the actual criteria themselves.

3.1. *Pruning During Recursive Generation.* We assume familiarity with the technique of backtracking as explained, for instance, in Bitner and Reingold [2].

We generate tables systematically with a recursive procedure, starting from an initially "all zero" table and filling more and more boxes with each possible nonzero value. We backtrack each time we "know" (on the basis of an easy criterion) that the table will be nonassociative no matter how we proceed (see 2.c), and after exploring all possibilities generated in the "lower" recursive calls. The method resembles the recursive generation of all table-codes as 4-ary numbers in ascending order.

For pruning the recursion (as indicated) we need a fast but powerful criterion to conclude nonassociativity on the generated part of a table. The trick is that once we know that the current part of a table is nonassociative we do not want to waste time on the explicit generation of its supertables, because they will come out to be nonassociative also (use 2.c). We decided to look for "trivial" contradictions of the form

$$v_i(v_jv_k) \neq (v_iv_j)v_k$$

among defined products.

The method eliminated as many as 230,355 tables as being provably nonassociative, using about 236 seconds of computing time. This simple criterion enabled us to eliminate about 88% of all tables!

The 31,789 tables for which the criterion failed were saved on a file.

3.2. *Elimination of Subtables of Order 3-Semigroups.* Among the 31,789 tables left there are 113 full tables (without *'s). These tables must be associative (as they passed the test in 3.1), and are easily recognized to be the 113 semigroups of order 3 (see, e.g. Plemmons [13]). By 2.b we can eliminate the semigroup tables and their subtables from the file.

As different semigroup tables may very well have many identical subtables, a straightforward elimination-routine could waste much time in cancelling tables "more than once". Our recursive generation and successive elimination of subtables was made more efficient by applying a simple pruning criterion again, based on the fact that once a subtable is found which was eliminated earlier then all of its subtables must have been eliminated earlier also (and, consequently, there is no need to go through them again).

The method eliminated 22,761 tables in about 46 seconds, reducing the number of tables left for consideration in the file to 9028.

3.3. *Table-Completion.* The remaining tables have no contradiction of the form $v_i(v_jv_k) \neq (v_iv_j)v_k$, but cannot be completed to an order 3 semigroup either.

After inspecting a number of tables by hand, we concluded that it would be mpractical to consider parenthesized products of size 4, 5, ... at this stage because he work required for each product of size 5 or larger would become too costly in computer time, while our estimates indicated that the number of tables eliminated by this method would remain relatively small.

We decided to consider all $v_i(v_jv_k) = (v_iv_j)v_k$ expressions again, to see which gave a one-letter value on one side and an "undefined" two-letter product on the other side. In this manner we obtain a completion of the table, where we note that the associativity problem of any completion remains equivalent to the associativity problem of the original table (see 2.d).

An elimination-criterion is obtained by observing that we may discard a table as nonassociative once we find contradictory products during the completion-process. Our routine was designed so as to eliminate all supertables of any such table also.

The method eliminated 7056 tables, leaving only 1972 tables yet undecided. The 7056 tables we eliminated were saved in a separate file for later inspection of "special properties".

3.4. *Forming a Conjecture.* We showed by hand that the first dozen tables left all were associative. Based on this remarkable discovery we conjectured at this point that all remaining tables might be associative (which will indeed turn out to be true).

Note that we may not expect that such a conjecture holds already at this same point if we were to do the current process on tables of any order greater than 3.

3.5. *Isomorphism Rejection.* The proofs of associativity for the first few tables varied widely, with no single heuristic clearly emerging as "the best". With a manageable number of tables left we decided to search for isomorphism classes of tables under an appropriate group of symmetries, and to apply the remaining tests to one representative of each class only.

The associativity problem for a table is invariant under any renaming of v_1, v_2, and v_3 (as implied by any permutation of these letters), and under mirroring around the main diagonal (which merely "reverses" the direction of the products). A class may contain up to 12 isomorphic tables.

For each uncancelled table we determined the representative in its class with smallest code-number, and cancelled all other tables in the class (which includes the table we started from, unless it happened to be the chosen representative) from the file.

The method reduced the number of tables to 193.

3.6. *Table-Completion.* Looking at a sample, it appeared that many tables had a same completion.

Using a previous procedure (see 3.3), we determined for each table as good a completion as we could get, and actually replaced each table in the file by its completion.

After eliminating duplicates and another round of isomorphism rejection we were left with only 31 tables.

3.7. *Elimination of Subtables.* By visual inspection we found that 15 tables were subtables of some other tables in the remaining set. Using 2.b and with the associativity conjecture (3.4) still unrefuted, we decided that we could eliminate all subtables for the remaining process. The resulting file of 16 tables is shown in Figure 1. We shall refer to the individual tables as 1-a, 1-b, ...

*	*	v_2
*	v_3	*
v_2	*	v_1

a

*	*	v_2
*	v_1	*
v_2	*	v_2

b

*	*	v_2
*	v_3	*
v_2	*	v_2

c

*	v_3	v_2
v_3	*	v_1
v_2	v_1	*

d

v_1	*	v_3
v_2	v_3	*
v_3	*	v_1

e

v_1	v_2	v_3
v_2	v_1	*
v_3	*	v_1

f

v_1	v_2	v_3
v_2	v_1	*
v_3	*	v_2

g

v_2	*	*
*	v_3	*
*	*	v_1

h

*	*	v_1
v_3	v_3	v_2
v_2	v_2	v_3

i

v_1	*	v_1
v_2	v_3	v_2
v_3	v_2	v_3

j

v_1	*	v_1
v_3	v_2	v_3
v_3	v_2	v_3

k

v_2	*	*
*	v_1	v_1
*	v_1	v_1

1

v_1	v_1	v_1
v_1	v_3	*
v_1	*	v_2

m

*	*	v_1
*	*	v_2
v_2	v_1	*

n

*	v_2	v_3
v_3	v_2	v_3
v_2	v_2	v_3

o

v_1	*	v_3
*	v_3	v_3
v_3	v_3	v_3

p

FIGURE 1

3.8. *Parikh Test.* For each chain $X = X_0 \xrightarrow[c,e]{} \cdots \longrightarrow X_i \longrightarrow \cdots \xrightarrow[c,e]{} X_n = Y$ one may keep track of the changing set of symbols of each intermediate string X_i in a so-called Parikh-vector $(\pi_1^{(i)}, \pi_2^{(i)}, \pi_3^{(i)})$ (as in Parikh [12], see also Salomaa [15]), where $\pi_j^{(i)}$ counts the number of symbols v_j in X_i.

In an expansion $v_k \longrightarrow v_i v_j$ we must lower the v_k-count by 1 and increment the v_i- and v_j-counts by 1 each. In a contraction $v_i v_j \longrightarrow v_k$ we must update the Parikh-vector in the opposite way. It follows that in each step the Parikh-vector changes by adding or subtracting one out of finitely many possible step-vectors $\delta_1, \ldots, \delta_t$ which correspond to the defined entries of the table.

If (say) $v_1 \overset{*}{\underset{c,e}{\longrightarrow}} v_2$, then there must be an integer solution $\lambda_1, \ldots, \lambda_t$ to the equation

$$(1, 0, 0) - \sum_{1}^{t} \lambda_S \delta_S = (0, 1, 0)$$

(but the converse is not necessarily true!). If the equation has no solution in integers, then we may conclude that $v_1 \not\equiv_T v_2$.

Similarly testing solvability of the resulting equations for each choice of v_i and v_j $(i \neq j)$ is the essence of the Parikh test for a table. If no equation is found to have a solution in integers, then we can discard the table as associative.

Example. The step-vectors for Table 1-a are

$$\begin{aligned}\delta_1 &= (-1, 0, 2),\\ \delta_2 &= (1, -1, 1),\\ \delta_3 &= (0, 2, -1).\end{aligned}$$

The equation

$$(1, 0, 0) + \lambda_1(-1, 0, 2) + \lambda_2(1, -1, 1) + \lambda_3(0, 2, -1) = (0, 1, 0)$$

has the unique solution $\lambda_1 = 2/5$, $\lambda_2 = -3/5$, $\lambda_3 = 1/5$. It follows that $v_1 \not\equiv_{1-a} v_2$. In a similar manner one can show that $v_1 \not\equiv_{1-a} v_3$ and $v_2 \not\equiv_{1-a} v_3$.

Applying the Parikh-test by hand, one can eliminate Tables 1-a through 1-h successfully as associative.

3.9. *Biased Parikh-Test.* One can observe that for each of the Tables 1-i to 1-k the following special property holds: whenever a product begins with the letter v_1, then any contraction or expansion of the product must begin with the letter v_1 also. It follows immediately that $v_1 \not\equiv v_2$ and $v_1 \not\equiv v_3$, and we only need to test the pair v_2, v_3. Using the Parikh-test, one can easily verify that $v_2 \not\equiv v_3$ for each of the tables, and we can eliminate 1-i through 1-k as associative.

In Table 1-l we can make the same observation for v_3, and we need not consider this letter further. The Parikh-test easily shows that $v_1 \not\equiv v_2$, and we conclude that 1-l is associative also.

For Table 1-m one can make the similar observation that whenever a product contains a v_1 (not necessarily up front), then any contraction or expansion of the product must also contain a v_1. Using the Parikh-test, one can show that $v_2 \not\equiv v_3$ again, and it follows that 1-m can be eliminated as associative.

3.10. *Tough Tables.* Tables 1-n through 1-p (the remaining cases) appear to be more difficult to prove associative. We shall develop a useful, but cumbersome technique in which we shall use the common notation for regular languages in formal language theory (see Salomaa [15]).

For each letter v_i we shall want to determine a set $R(v_i)$ with the following properties:

(a) $v_i \in R(v_i)$,

(b) if $X \in R(v_i)$ and $X \underset{c,e}{\rightarrow} Y$, then $Y \in R(v_i)$.

An $R(v_i)$ will always contain $\{X | v_i \overset{*}{\underset{c,e}{\rightarrow}} X\}$ (the equivalence class of v_i), but we do not demand that each $R(v_i)$ necessarily coincides with this set (i.e., we do not demand that $R(v_i)$ is as small as possible). The practical reason is that the exact equivalence class of a v_i may be hard to determine, and that it may be easier to describe a wider set which is closed under the expansion and contraction operations.

To test for associativity it is sufficient to compute just two $R(v)$-sets and to check that $v_i \notin R(v_j)$ for all i and (the two relevant values for) j, with $i \neq j$.

Tables 1-n through 1-p are proved to be associative by taking the following sets, respectively,

$$\begin{aligned}\text{1-n: } R(v_1) &= (v_3v_3)^*v_1v_3^* \cup (v_3v_3)^*v_3v_2v_3^*,\\ R(v_2) &= (v_3v_3)^*v_2v_3^* \cup (v_3v_3)^*v_3v_1v_3^*,\end{aligned}$$

$$\begin{aligned}\text{1-o: } R(v_2) &= (v_1^*v_2^*v_3^*)^*v_2(v_1v_1)^* \cup (v_1^*v_2^*v_3^*)^*v_3v_1(v_1v_1)^*,\\ R(v_3) &= (v_1^*v_2^*v_3^*)^*v_3(v_1v_1)^* \cup (v_1^*v_2^*v_3^*)^*v_2v_1(v_1v_1)^*,\end{aligned}$$

$$\begin{aligned}\text{1-p: } R(v_1) &= v_1^*,\\ R(v_2) &= v_2.\end{aligned}$$

It follows that we have finished the classification of all tables of order 3. We proved

THEOREM 3.a. *The associativity problem is effective for all tables of order* $\leqslant 3$.

3.11. *Tables of Order* 2. Along the way we must have encountered all tables which are "essentially" of order 2. By changing some parameters in our routines we can quickly run the analysis again, this time for the 81 tables of order 2 only.

The "pruning during recursive generation" eliminates 16 tables with a contradiction of the form $v_i(v_jv_k) \neq (v_iv_j)v_k$. Among the remaining tables there are 8 full tables, which must be semigroups! Some of the full tables are isomorphic, and it appears that there are only 4 "distinct" semigroups of order 2 (see Figure 2).

v_1	v_1
v_1	v_1

v_1	v_1
v_1	v_2

v_1	v_1
v_2	v_2

v_1	v_2
v_2	v_1

FIGURE 2

It turns out that the "elimination of subtables of order 2 semigroups" removes as many as 64 of the remaining 65 tables as associative from the file. The one table left to consider is shown in Figure 3. This table is easily shown to be associative with the Parikh-test.

v_2	*
*	v_1

FIGURE 3

4. Some Statistics for Order 3 Tables. Among the 262,144 tables of order 3 we found 24,733 associative tables (about 9%) and 237,411 nonassociative tables.

Note that among the 81 tables of order 2 we found 65 associative tables (about 80%) and only 16 nonassociative tables. It is interesting that in this case the number of associative tables is still relatively large. It is likely that the percentage decreases very rapidly for larger order.

Among the 237,411 nonassociative tables of order 3 we found as many as 230,355 tables to be nonassociative because of a direct contradiction of the form $v_i(v_jv_k) \neq (v_iv_j)v_k$. The remaining 7056 tables were proved to be nonassociative in an attempted completion. As they were saved on a separate file, we decided to examine these tables in detail to see how much "deeper" the contradictions were hidden. We shall first look for contradictions in expressions of a certain size by just choosing different bracketings (which essentially gives one-mountain contradictions).

4.1. *Contradictions by Different Bracketings.* Among the 7056 tables we spotted 6072 tables with a contradiction on a size 4 expression (only 572 nonisomorphic cases), 492 tables with a contradiction on a size 5 expression (only 42 nonisomorphic cases), and 96 more tables with a contradiction on a size 6 expression (only 8 nonisomorphic cases).

Example. Table 4-a (Figure 4) has no size 3 contradiction, but it has a size 4 contradiction

$$\{v_1 =\}(v_3(v_2v_3))v_3 \neq v_3(v_2(v_3v_3))\{= v_2\}.$$

Table 4-b has no size 3 or 4 contradictions, but it has a size 5 contradiction

$$\{v_1 =\}(v_3(v_2(v_3v_2)))v_3 \neq (v_3v_2)(v_3(v_2v_3))\{= v_3\}.$$

Table 4-c has no size 3, 4, or 5 contradictions, but it has a size 6 contradiction

$$\{v_1 =\}((v_3(v_3v_2))(v_3v_2))v_2 \neq v_3((v_3((v_2v_3)v_2))v_2)\{= v_3\}.$$

*	*	*
*	*	v_1
v_2	*	v_3

a

*	*	*
*	*	v_1
v_2	v_3	*

b

*	*	*
*	*	v_3
v_3	v_1	*

c

FIGURE 4

The amount of computer time spent per table now became a strong argument not to continue the search for contradictions on expressions of a larger size in the remaining tables.

We conclude that there are 396 tables left which have no contradictions of size less than or equal to 6, and which would require us to look at expressions of size 7 or larger or which would require a chain with "several mountains" (rather than just one) to prove a contradiction.

4.2. *Tables With No One-Mountain Contradiction.* We reduced the 396 remaining tables to 36 nonisomorphic cases, and made a further reduction by observing that each of the 36 tables was in fact a subtable of one of the 4 tables in Figure 5.

v_1	*	v_1
v_1	v_1	v_1
v_1	v_2	v_1

a

v_1	*	*
*	v_1	v_3
v_1	v_1	v_1

b

v_1	*	*
*	v_2	v_1
v_3	*	v_1

c

v_1	*	*
v_2	v_1	v_1
v_3	v_1	v_1

d

FIGURE 5

We shall prove that none of these tables (and thus none of the 396 tables considered here) has a single mountain contradiction. This proves an earlier claim (see 2.1) that Tamari's one-mountain theorem is not valid for general tables, which was demonstrated before only with a single example of order 6.

We shall consider the tables in Figure 5 one-by-one.

In Table 5-a it appears that $v_1 \equiv v_2$, for example by means of the following chain

$$v_2 \underset{e}{\to} v_3v_2 \underset{e}{\to} v_3v_3v_2 \underset{c}{\to} v_1v_2 \underset{e}{\to} v_2v_2v_2 \underset{c}{\to} v_2v_1 \to v_1.$$

It is not hard to argue that there can be no single mountain chain from v_2 to v_1. The letter v_2 can be expanded into strings of the form $v_3v_3^*v_2$ only, whereas subsequent application of contractions on any such string can never eliminate the v_2 at the end.

In Table 5-b one can rename v_2 as v_3 and v_3 as v_2, resulting in a table which is isomorphic to a subtable of 5-a. The nonvalidity of the one-mountain theorem follows immediately from the previous case.

For Tables 5-c and 5-d it is harder to prove the nonexistence of a one-mountain contradiction. We shall use a technique as in 3.10, involving the common notation for regular languages again (Salomaa [15]).

For each letter v_i we shall want to determine a set $S(v_i)$ with the following properties:

(a) $v_i \in S(v_i)$,

(b) if $X \in S(v_i)$ and $X \underset{e}{\to} Y$, then $Y \in S(v_i)$.

An $S(v_i)$ always contains $\{X | v_i \overset{*}{\underset{e}{\to}} X\}$, but we do not require that $S(v_i)$ necessarily coincides with this smallest possible choice. Note that $\{X | v_i \overset{*}{\underset{e}{\to}} X\}$ is always a context-free language, but we shall find it easier to describe a regular $S(v_i)$ which perhaps strictly includes this set.

To test the nonvalidity of the one-mountain theorem it is sufficient to compute an S-set for each v_i and to prove that $S(v_i) \cap S(v_j) = \varnothing$ for all i, j with $i \neq j$ (which would show, in fact, that there is no pair $v_i \equiv v_j$ at all with a one-mountain proof).

For Tables 5-c and 5-d it is accomplished with the following S-sets respectively:

$$\begin{aligned}
\text{5-c:}\quad & S(v_1) = ((v_2^*v_2v_3)^*(v_3v_3)^*v_1^*)^*,\\
& S(v_2) = v_2^+,\\
& S(v_3) = v_3((v_2^*v_2v_3)^*(v_3v_3)^*v_1^*)^*,
\end{aligned}$$

$$\begin{aligned}
\text{5-d:}\quad & S(v_1) = (v_1^*(v_2v_1^*v_3v_1^*)^*(v_2v_1^*v_2v_1^*)^*(v_3v_1^*v_3v_1^*)^*(v_3v_1^*v_2v_1^*)^*)^*,\\
& S(v_2) = v_2(v_1^*(v_2v_1^*v_3v_1^*)^*(v_2v_1^*v_2v_1^*)^*(v_3v_1^*v_3v_1^*)^*(v_3v_1^*v_2v_1^*)^*)^*,\\
& S(v_3) = v_3(v_1^*(v_2v_1^*v_3v_1^*)^*(v_2v_1^*v_2v_1^*)^*(v_3v_1^*v_3v_1^*)^*(v_3v_1^*v_2v_1^*)^*)^*.
\end{aligned}$$

This completes the analysis of nonassociative tables of order 3.

5. Acknowledgement. We thank Professor Patricia J. Eberlein for helpful suggestions in an early stage of the work.

Department of Computer Science
State University of New York at Buffalo
Amherst, New York 14226

Computer Science Department
The Pennsylvania State University
University Park, Pennsylvania 16802

Department of Mathematics
State University of New York at Buffalo
Amherst, New York 14226

1. H. H. BAKER, A. K. DEWDNEY & A. L. SZILARD, "Generating the nine-point graphs," *Math. Comp.*, v. 28, 1974, pp. 833–838.

2. J. R. BITNER & E. M. REINGOLD, "Backtrack programming techniques," *Comm. ACM*, v. 18, 1975, pp. 651–656.

3. R. H. BRUCK, *A Survey of Binary Systems*, Springer-Verlag, New York, 1966.

4. A. H. CLIFFORD & G. B. PRESTON, *The Algebraic Theory of Semigroups*, Math. Surveys, Vol. 7, Part I, Amer. Math. Soc., Providence, R. I., 1964.

5. J. B. DE CARVALHO & D. TAMARI, "Sur l'associativité partielle des symétrisations de semi-groupes," *Portugal. Math.*, v. 21, 1962, pp. 157–169.

6. G. E. FORSYTHE, "SWAC computes 126 distinct semigroups of order 4," *Proc. Amer. Math. Soc.*, v. 6, 1955, pp. 443–447.

7. H. JÜRGENSEN, "Calculation with the elements of a finite group given by generators and defining relations," in *Computational Problems in Abstract Algebra* (J. Leech, Editor), Pergamon Press, Oxford, 1970.

8. H. JÜRGENSEN, *Ueber das Rechnen mit den Elementen abstrakt präsentierter Halbgruppen*, Bericht 76/06, Inst. f. Informatik u. Praktische Math., Christian-Albrechts-Universität, Kiel, 1976.

9. H. JÜRGENSEN & P. WICK, "Die Halbgruppen der Ordnungen $\leqslant 7$," *Semigroup Forum*, v. 14, 1977, pp. 69–79.

10. E. S. LJAPIN, *Semigroups*, Transl. Math. Monographs, vol. 3, Amer. Math. Soc., Providence, R. I., 1963.

11. M. H. A. NEWMAN, "On theories with a combinatorial definition of equivalence," *Ann. of Math.*, v. 43, 1942, pp. 223–243.

12. R. J. PARIKH, "On contextfree languages," *J. Assoc. Comput. Mach.*, v. 13, 1966, pp. 570–581.

13. R. J. PLEMMONS, "There are 15973 semigroups of order 6," *Math. Algorithms*, v. 2, 1967, pp. 2–3.

14. R. J. PLEMMONS, "Construction and analysis of non-equivalent finite semigroups," in *Computational Problems in Abstract Algebra* (J. Leech, Editor), Pergamon Press, Oxford, 1970.

15. A. SALOMAA, *Formal Languages*, Academic Press, New York, 1973.

16. H. R. STRONG, A. MAGGIOLO-SCHETTINI & B. K. ROSEN, "Recursion structure simplification," *SIAM J. Comput.*, v.4, 1975, pp. 307–320.

17. D. TAMARI, "Imbeddings of partial (incomplete) multiplicative systems (monoids), associativity and word problem," *Notices Amer. Math. Soc.*, v. 7, 1960, p. 760; Abstract #575-30.

18. D. TAMARI, *Problèmes d'Associativité des Monoides et Problèmes des Mots pour les Groupes*, Séminaire Dubreil-Pisot, Vol. 16, 1962/1963, pp. 7.01–7.30.

19. D. TAMARI, *Le Problème d'Associativité des Monoides et le Problème des Mots pour les Demi-Groupes*, Séminaire Dubreil-Pisot, vol. 24, 1970/1971, pp. 8.01–8.15.

20. D. TAMARI, "The associativity problem for monoids and the word problem for semigorups and groups," in *Word Problems* (W. W. Boone, Editor), North-Holland, Amsterdam, 1973.

21. K. T. TETSUYA, T. HASHIMOTO, T. AKASAWA, R. SHIBOTA, T. INUI & T. TAMURA, "All semigroups of order at most 5," *J. Gakugei Tokushima Univ.*, v. 6, 1955, pp. 19–39.

22. R. B. WEISS, *Finding Isomorph Classes for Combinatorial Structures*, MAC Techn. Memo 64, M. I. T., Cambridge, Mass., 1975.