# THE FINITENESS OF FINITELY PRESENTED MONOIDS*

Robert McNaughton†
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180-3590, U.S.A.
mcnaught@cs.rpi.edu

April, 1997

**Introduction.** It is undecidable whether a monoid given by a finite presentation is finite (see, e.g., [1], pp. 157–160). On the other hand, with the mere knowledge that the monoid is finite, one can effectively construct the multiplication table of the monoid, and thus obtain a complete understanding of its structure. This paper will present this construction method in detail (Section 2) and offer some remarks about its computational complexity (Section 3). The notation here will be based on [1], whose final chapter furnishes much of the background and will be referenced frequently.

It is convenient to think of a finite monoid presentation as a Thue system, i.e., an ordered pair $(\Sigma, R)$, where $\Sigma$ is a finite alphabet (or set of generators) and $R$ is a finite set of unordered pairs of strings over $\Sigma$, called the "set of rules." For $(y_1, y_2) \in R$ and $x, z \in \Sigma^*$ one writes $xy_1z \leftrightarrow xy_2z$ and $xy_2z \leftrightarrow xy_1z$; thus the rules operate symmetrically. One writes $x \leftrightarrow^* y$ to assert the existence of a *derivation* of $y$ from $x$, i.e., a sequence

$$x_0 = x, x_1, \ldots, x_p = y \ (p \geq 0), \text{ where } x_i \leftrightarrow x_{i+1} \text{ for each } i \leq p-1$$

When $x \leftrightarrow^* y$ holds one says that $x$ and $y$ are *congruent*. For brevity, the $\leftrightarrow^*$ relation is written as $\equiv$. It is clearly a congruence relation, since it is reflexive, symmetric and transitive, and since, for all $x, y_1, y_2, z \in \Sigma^*$, $y_1 \equiv y_2$ implies $xy_1z \equiv xy_2z$.

*This paper is dedicated to the memory of Marco Schützenberger, the one most responsible for impressing upon me the importance of monoids.

†Supported by Grant No. CCR-9500182 from the National Science Foundation. I am grateful for the crucial guidance of Friedrich Otto in this research, and also for his critical reading of this paper.

For the presentation $P = (\Sigma, R)$, $M(P)$ is defined to be the monoid whose elements are the congruence classes of $\Sigma^*$ of the Thue system congruence $\equiv$. For $w \in \Sigma^*$, $[w]$ is defined to be the congruence class $\{x|x \equiv w\}$. Thereupon the set of elements of $M(P)$ is $\{[w] \mid w \in \Sigma^*\}$.

Monoid multiplication is defined as $[x][y] = [xy]$. Where $e$ is the null word, i.e., the word of length 0, $[e]$ is the monoid identity (since $[e][x] = [x][e] = [x]$, for all $x$). Since word concatenation is associative, multiplication in $M(P)$ is associative; so $M(P)$ is indeed a monoid.

For $x \in \Sigma^*$, $|x|$ is the length of $x$. For $M$ a monoid, $|M|$ is the order of $M$.

**Theorem 1.1.** If $x \in \Sigma^*$ and $x$ is not congruent modulo $P$ to any shorter word then $|M(P)| \geq |x| + 1$.

*Proof:* If $x = x'x''x'''$ with $|x''| > 0$ and $x' \equiv x'x''$ then $x \equiv x'x'''$ where $|x'x'''| < |x|$. From this it follows that if $x$ is not congruent to any shorter word then no two prefixes of $x$ are congruent. Since $x$ has $|x| + 1$ prefixes, including $e$ and $x$ itself, there are at least $|x| + 1$ congruence classes, viz., $|M(P)| \geq |x| + 1$.□

We offer two examples with $\Sigma = \{b, c\}$. For $P_1 = (\Sigma, \{(bc, e)\})$, $M(P_1)$ turns out to be an infinite monoid, known as the "bicyclic monoid" or "bicyclic semigroup," whose elements are of the form $[c^i b^j]$, $i, j \geq 0$, where $[e = c^0 b^0]$ is the monoid identity and the product is

$$[c^i b^j][c^k b^m] = \begin{cases} [c^{i+k-j} b^m] & \text{if } j \leq k \\ [c^i b^{m+j-k}] & \text{if } j > k \end{cases}$$

(There are some who might want to regard this expression as a multiplication table. However, this paper will consider multiplication tables only for finite monoids.)

On the other hand, for the example

$$P_2 = (\Sigma, \{(bbb, bb), (cc, c), (bc, c), (cbb, bb)\})$$

$M(P_2)$ is finite. The five monoid elements are the five congruence classes $[e] = \{e\}$, $[b] = \{b\}$, $[bb] = \{\Sigma^* bb\}$, $[c] = \{\Sigma^* c\}$ and $[cb] = \{\Sigma^* cb\}$. The multiplication table is depicted here with one column for each monoid element but with rows corresponding only to the monoid generators. In the following table for $M(P_2)$, each entry is the result of the product of the column element (left factor) by the row element (right factor):

| | $[e]$ | $[b]$ | $[bb]$ | $[c]$ | $[cb]$ |
|---|---|---|---|---|---|
| $b$ | $[b]$ | $[bb]$ | $[bb]$ | $[cb]$ | $[bb]$ |
| $c$ | $[c]$ | $[c]$ | $[c]$ | $[c]$ | $[c]$ |

The entire multiplication table could easily be written from this abbreviated table; e.g.,

$$[cb][cb] = ([cb]c)b = [c]b = [cb]$$

The problem of the finiteness of finitely presented monoids is somewhat similar to the problem of whether a given Thue system has an equivalent (finite) confluent system, with respect to a given ordering of strings. Indeed the Knuth-Bendix completion procedure ([3], see also [2]) is a semi-algorithm for this problem: it terminates with an equivalent confluent system if there is one, but fails to terminate if there is none. Furthermore, the Knuth-Bendix procedure could be used as a semi-algorithm for the finiteness problem of finitely presented monoids, as will be explained at the end of Section 2.

The Todd-Coxeter procedure [6], which has as its goal the enumeration of the cosets of a subgroup in a group, can be used to construct the multiplication table of a finite group from a given finite presentation. It is quite similar to the algorithm that will be given in the next section, which could be thought of as its generalization to monoids.

The Todd-Coxeter procedure begins with a subgroup $H$ of a given finitely presented group $G$, enumerates the cosets of $H$ in $G$, and produces a multiplication table containing all products of the form $H_ib = H_j$, where $H_i$ and $H_j$ are cosets of $H$ and $b$ is a generator of $G$. One way of using the Todd-Coxeter procedure to find the multiplication table of a group $G$ is simply to take $H$ as the trivial subgroup of $G$, whereupon the cosets of $H$ are the singleton sets, identifiable with the elements of $G$.

Coset enumeration in general, including the Todd-Coxeter procedure in particular, plays a prominent rôle in computational group theory (see, e.g., [5], especially, pp. 175–196). A proof that the procedure does terminate and produce all the cosets, in cases where the subgroup $H$ is of finite index in $G$, can be found in [4].

In the remainder of this paper the word "presentation" will always mean a finite monoid presentation.

## 2. The construction.

This section puts forth a procedure that, from a given presentation, terminates if and only if the monoid is finite; upon termination it yields the multiplication table of the monoid. Let $T = (\Sigma, R)$ be the presentation, i.e., a Thue system as explained in Section 1. We require a refinement of the congruence relation $\equiv$:

**Definition:** $x \equiv_n y$ means that $|x| \leq n$, $|y| \leq n$ and there is derivation of $y$ from $x$ in $(\Sigma, R)$ in which no word has length greater than $n$. Note that (1) the 3-place predicate $x \equiv_n y$ is decidable. For each $n$, (2) the relation $\equiv_n$ may be thought of as a congruence relation over $(\Sigma \cup \lambda)^n$ but (3) it is certainly not a congruence relation over $\Sigma^*$.

**Definition:** $D(k, n)$ is the following predicate over the positive integers: for every word $w$ of length $k$, there is a word $w'$ of smaller length such that $w \equiv_n w'$. Note that $D(k, n)$ implies $k \leq n$, and that $D$ is a decidable predicate.

The reader is also asked to verify that it is decidable whether, for any given $n$, there exists $k$ such that $D(k, n)$.

**Theorem 2.1.** $M(P)$ is finite if and only if there exist $k$ and $n$ such that $D(k, n)$ holds.

*Proof:* Assume $D(k, n)$. Since every word of length $k$ is congruent to a word of smaller length, every word of length $\geq k$ is congruent to a shorter word. From this it follows that every word of length $\geq k$ is congruent to a word of length less than $k$. Thus $M(P)$ is finite.

For the converse, assume $M(P)$ is finite. For each $x \in \Sigma^*$, let $h(x) =$ the length of the shortest word congruent to $x$. The set $H = \{h(x) \mid x \in \Sigma^*\}$ is bounded, since $M(P)$ is finite. Take $k = 1 +$ the maximum element of $H$.

For each $y \in \Sigma^k$, there is a derivation from $y$ to a shorter word. Since there are finitely many such $y$ there are finitely many such derivations, each of which is a finite sequence of words. Thus the set of all words occurring in all the derivations has a maximum length $n$. For these values of $k$ and $n$, $D(k, n)$ holds.$\Box$

**Definition:** $n_0 =$ the smallest value of $n$ such that $D(k, n)$ is true for some $k$; it is undefined if $M(P)$ is infinite.

In order to search for $n_0$, we construct tables for the relations $\equiv_1, \ \equiv_2, \ \cdots$ such that the entries for $\equiv_n$ are all words of length $\leq n$. The table for $\equiv_{n+1}$ is readily constructed from the table for $\equiv_n$. Indeed, if we take

$$Q_{n+1} = \{< x, y > \mid |x| = n + 1, |y| \leq n + 1, x \leftrightarrow y\}$$

then $\equiv_{n+1}$ is the symmetric and transitive closure of the relation

$$Q_{n+1} \ \cup \ \equiv_n \ \cup \ \{< x, x > \mid |x| = n + 1\}$$

For the remainder of this section, we assume that $n_0$ has been found and therefore $M(P)$ is finite. Since every word of length $n_0$ is congruent to a word of smaller length, every word of length $\geq n_0$ is congruent to a word of length less than $n_0$.

An often-used total ordering of all words of $\Sigma^*$ is the *length-lexicographic ordering*. We shall use the symbol $<$ for this ordering, writing $w_1 < w_2$ $(w_1, w_2 \in \Sigma^*)$ if either $|w_1| < |w_2|$ or else $|w_1| = |w_2|$ and $w_1$ precedes $w_2$ lexicographically. For example, if $b, c, d \in \Sigma$ with $b < c$ and $c < d$, then $bcc < bdb$ and $dd < bbb$.

**Definition:** For $|x| \leq n_0$, let $f(x) =$ the smallest word $w$ in the sense of the length-lexicographic ordering relation such that $w \equiv_{n_0} x$. Note that, for all such $x$, $|f(x)| \leq n_0 - 1$. Let the range of $f$ (which is finite) be $\{w_1 = e, w_2, \ldots, w_p\}$.

Form the directed graph $G_1$ as follows: The nodes are labeled $w_1, w_2, \ldots, w_p$; $w_1 = e$ is called the "initial node." For all $w_i, w_j$ and $b$, there is an arc labeled $b$ from $w_i$ to $w_j$ if $f(w_i b) = w_j$; there are no arcs other than these.

**Definition:** We write $\delta(w_i, x) = w_j$ if there is a walk in $G_1$ from node $w_i$ to node $w_j$ whose labels in order spell out the word $x$. For $x = e$, $w_j = w_i$ and the walk is the null walk on $w_i$.

**Theorem 2.2.** For all $x \in \Sigma^*$, there is a walk in $G_1$ spelling out $x$ from $e$ to $\delta(e, x)$.

*Proof:* First note that, for $b \in \Sigma$, if node $N$ has label $w$ then $|w| < n_0$, $|wb| \leq n_0$ and $|f(wb)| < n_0$. So there is an arc labeled $b$ from node $w$ $(= N)$ to node $f(wb)$. From this our theorem follows by induction on the length of $x$ (the basis being the null walk on $e$).$\square$

**Theorem 2.3.** For all $x \in \Sigma^*$, $\delta(e, x) \equiv x$.

*Proof by induction on $|x|$:* Clearly our theorem is true for $x = e$. Assume $w_i = \delta(e, x) \equiv x$. For $b \in \Sigma$, we have $\delta(e, xb) = \delta(w_i, b) = w_j$, where $w_j = f(w_i b)$. By definition of $f$, $w_j \equiv_{n_0} w_i b$, which implies $w_j \equiv w_i b$. Thus

$$\delta(e, xb) = w_j \equiv w_i b \equiv xb \text{ (since } w_i \equiv x).\square$$

**Corollary.** For all $x, y \in \Sigma^*$, if $\delta(e, x) = \delta(e, y)$ then $x \equiv y$.

For $|x|, |y| \leq n_0$, $x \equiv_{n_0} y$ implies $x \equiv y$. However, the converse may not hold, as shown in Example 1 at the end of this section. Consequently, the graph $G_1$ may not have the converse to the property of the last Corollary, which we need. In general, therefore, the graph $G_1$ needs to be simplified to a graph that has both properties. Accordingly, a sequence of graphs, $G_2, \cdots, G_p$ will be constructed so that $G_p$ will have the following properties:

(1) if $\delta(e, x) = \delta(e, y)$ in $G_p$ then $x \equiv y$;

(2) if $x \equiv y$ then $\delta(e, x) = \delta(e, y)$ in $G_p$; and

(3) $G_p$ is deterministic.

The graph $G_1$ has properties (1) and (3). It does not have property (2) unless $p = 1$. As we shall see, $G_2, \ldots G_{p-1}$, which may or may not have property (3), all have property (1) but not property (2). For $1 \leq i \leq p-1$, $G_{i+1}$ is obtained from $G_i$ by merging two nodes, as will now be prescribed:

**Definition:** For any node $N$ in a graph $G$, $L_G(N) = \{x | x$ is spelled out by a walk from the initial node to $N$ in $G\}$. Note that if there is an arc labeled $b$ from node $N$ to node $N'$ in $G$ then $L_G(N)b \subseteq L_G(N')$. ($L_G(N)b$ means $\{wb | w \in L_G(N)\}$.)

**Definition:** The nodes $N_1$ and $N_2$ of a graph are an *eligible pair for merging* if $N_1 \neq N_2$ and either (1) there are a node $N_4$, a letter $b$, an arc from $N_4$ to $N_1$ labeled $b$ and an arc from $N_4$ to $N_2$ labeled $b$, or else (2) there are a node $N_4$, a rule $(u, v)$, a walk from $N_4$ to $N_1$ spelling out $u$ and a walk from $N_4$ to $N_2$ spelling out $v$. ($N_4$ may be the same as $N_1$ or $N_2$, or may differ from each.)

For each $i$, $G_{i+1}$ is obtained from $G_i$ by merging some eligible pair of nodes. When node $N_1$ and $N_2$ in $G_i$ are merged, they are replaced by a single node $N_3$ in $G_{i+1}$; all arcs entering either $N_1$ or $N_2$ now enter $N_3$, and all arcs leaving $N_1$ or $N_2$ now leave $N_3$. Other nodes and arcs in $G_{i+1}$ are the same as in $G_i$. The initial node of $G_i$, merged or unmerged, becomes the initial node of $G_{i+1}$.

**Theorem 2.4.** For every $G_i$ and $x \in \Sigma^*$, there is at least one node $N$ such that $x \in L_{G_i}(N)$.

*Proof by induction on i:* The truth for $G_1$ follows from Theorem 2.2. By construction, if it holds for $G_i$ it holds for $G_{i+1}$.$\square$

**Theorem 2.5.** (1) For any node $N$ of any $G_i$, the words in the set $L_{G_i}(N)$ are all congruent to one another. And (2) for any two nodes $N_1$ and $N_2$ eligible for merging, the words in the set $L_{G_i}(N_1) \cup L_{G_i}(N_2)$ are all congruent to one another.

*Proof:* By the Corollary to Theorem 2.3, (1) is true of $G_1$. The proof of our theorem will be by mathematical induction on $i$ and the strategy will be to prove first Part a: if (1) holds for $i$ then (2) holds for $i$; and then Part b: if (1) and (2) hold for $i < p$ then (1) holds for $i + 1$.

*Part a:* Assume (1) is true for $G_i$ and let $N_1, N_2 \in G_i$ be eligible for merging. Case I: for some $b \in \Sigma$, there are arcs labeled $b$ from some node $N_4$ to $N_1$ and from $N_4$ to $N_2$. Let $y \in L_{G_i}(N_4)$. Then $yb \in L_{G_i}(N_1)$ and $yb \in L_{G_i}(N_2)$. From this fact and (1) for $G_i$, it follows that all words of $L_{G_i}(N_1) \cup L_{G_i}(N_2)$ are congruent to one another.

Case II: for some node $N_4$ of $G_i$, there is a walk from $N_4$ to $N_1$ spelling out $u$ and a walk from $N_4$ to $N_2$ spelling out $v$, where $(u, v)$ is a rule. Then $yu \in L_{G_i}(N_1)$ and $yv \in L_{G_i}(N_2)$. Since $yu \equiv yv$ and (1) is true for $G_i$, all words of $L_{G_i}(N_1) \cup L_{G_i}(N_2)$ are congruent to one another.

*Part b:* Assume (1) and (2) are true of $G_i$. For every node $N \in G_i$, let $con_i(N)$ be the common congruence class of the words spelled out by walks in $G_i$ from the initial node to $N$. We define a function $con'_{i+1}$ over the nodes of $G_{i+1}$:

$$con'_{i+1}(N) = \begin{cases} con_i(N) & \text{if } N \neq N_3 \\ con_i(N_1) = con_i(N_2) & \text{if } N = N_3 \end{cases}$$

**Lemma b1.** If there is an arc labeled $b$ from $N$ to $N'$ in $G_{i+1}$ then $con'_{i+1}(N') = [wb]$, where $w$ is any word in $con'_{i+1}(N)$.

*Proof:* Case I: $N \neq N_3 \neq N'$. Then $N, N'$ and the arc joining them are in $G_i$; and $con'_{i+1}(N) = con_i(N)$ and $con'_{i+1}(N') = con_i(N') = [wb]$ for any $w \in con_i(N)$.

Case II: $N = N_3 \neq N'$. Then there is an arc in $G_i$ labeled $b$ either from $N_1$ to $N'$ or from $N_2$ to $N'$; without loss of generality assume the former. The set $con'_{i+1}(N') = con_i(N') = [wb]$ for any $w \in con_i(N_1) = con'_{i+1}(N_3)$.

The proofs of the remaining two cases are left to the reader:

Case III: $N \neq N_3 = N'$.

Case IV: $N = N_3 = N'$. (Note that, in this case, there is an arc in $G_i$ labeled $b$ either from $N_1$ to $N_1$, from $N_1$ to $N_2$, from $N_2$ to $N_1$ or from $N_2$ to $N_2$.)□

**Lemma b2.** For all nodes $N$ of $G_{i+1}$, $L_{G_{i+1}}(N) \subseteq con'_{i+1}(N)$.

The proof consists of proving by use of Lemma b1 that, for any walk $W$ in $G_{i+1}$ from the initial node to $N$, if $W$ spells out $w \in \Sigma^*$ then $w \in con'_{i+1}(N)$. This proof, which is by induction on the length of the walk $W$, is left to the reader.

Lemma b2 completes part (b) and, with it, the entire proof of Theorem 2.5.□

**Theorem 2.6.** (1) The sequence $G_1, G_2, \ldots$ is finite. That is to say, there exists a $p$ such that no two nodes of $G_p$ are eligible for merging. (2) $G_p$ is deterministic.

*Proof:* For (1), note that the number of graph nodes in each graph in the sequence decreases by 1 each time. For (2) assume $G_i$ is nondeterministic. Then for some node $N_3$ of $G_i$ and some letter $b$, $\delta(N_3, b)$ has at least two values in $G_i$, say $N_1$ and $N_2$, eligible for merging. Hence, $i \neq p$.□

**Theorem 2.7.** If in $G_i$, $N_1 \neq N_0$, $x \in L_{G_i}(N_1)$, $y \in L_{G_i}(N_0)$ and $x \equiv y$, then $G_i$ has a pair of nodes eligible for merging.

*Proof:* If for some node $N_4$ of $G_i$ there is a letter $b$ with arcs from $N_4$ labeled $b$ to two distinct nodes, then these two nodes are eligible for merging. So let us assume that there is no such node $N_4$, viz., $G_i$ is deterministic. Thus, for any node $N$ of $G_i$ and word $w$, the node $\delta(N, w)$ is unique.

Where $N_1, N_0, x$ and $y$ are as in the hypothesis of our theorem, we have $x \neq y$. Let $z_1, z_2, \ldots, z_q$ be a derivation such that $x = z_1$ and $y = z_q$. By Theorem 2.4, for each $j$, $1 \leq j \leq q$, there is a node $N_j$ such that $z_j \in L_{G_i}(N_j)$. Knowing that $N_1 \neq N_0 = N_q$, we take $h$ to be the smallest positive integer such that $N_h \neq N_{h+1}$. Since $z_h \leftrightarrow z_{h+1}$ and $z_h \neq z_{h+1}$, we have $z_h = sut$ and $z_{h+1} = svt$, where $(u, v)$ or $(v, u)$ is a rule. Put $N'_4 = \delta(e, s)$. Then

$$\delta(N'_4, ut) = N_h \neq N_{h+1} = \delta(N'_4, vt)$$

Put $N'_1 = \delta(N'_4, u)$ and $N'_2 = \delta(N'_4, v)$, so $\delta(N'_1, t) = N_h$ and $\delta(N'_2, t) = N_{h+1}$. Since $N_h \neq N_{h+1}$, it must be that $N'_1 \neq N'_2$ and the pair $(N'_1, N'_2)$ is eligible for merging.□

**Corollary.** For $N_1 \neq N_2$, $x \in L_{G_p}(N_1)$ and $y \in L_{G_p}(N_2)$, $x$ and $y$ are not congruent.

**Main Theorem.** If $M(P)$ is finite, the elements of the monoid are represented one-to-one by the nodes of the graph $G_p$, from which the multiplication table can be written readily.

*Proof:* Theorem 2.5(1) and the Corollary to Theorem 2.7.□

It is possible that a generator may turn out to be congruent either to another generator or to $e$. If that happens the monoid can be thought of as having fewer generators than were called for in the presentation. However, general simplification of the generator set is not discussed in this paper. Nor is the problem of whether the monoids of given distinct multiplication tables are isomorphic.

**Example 1:** $P = (\{b, c\}, \{(bb, b), (cc, c), (bcb, b), (cbc, c), (b^5, c^5)\})$. The nontrivial equivalence classes for $\equiv_2$ are $\{b, bb\}$ and $\{c, cc\}$. Since neither $bc$ nor $cb$ is related by $\equiv_2$ to a shorter word, we see that $n_0 > 2$. But the nontrivial equivalence classes for $\equiv_3$ are $\{b, bb, bbb, bcb\}$, $\{c, cc, ccc, cbc\}$, $\{bc, bbc, bcc\}$ and $\{cb, cbb, ccb\}$. Every word of length 3 is related by $\equiv_3$ to a word of shorter length, so $n_0 = 3$. The graph $G_1$ is given by the following table, in which each monoid element is represented by its minimal word according to the $<$ relation defined above. (The brackets as used in the table of Section 1 are omitted here.)

|  | $e$ | $b$ | $c$ | $bc$ | $cb$ |
|---|---|---|---|---|---|
| $b$ | $b$ | $b$ | $cb$ | $b$ | $cb$ |
| $c$ | $c$ | $bc$ | $c$ | $bc$ | $c$ |

Since $\delta(e, b^5) = b$ and $\delta(e, c^5) = c$, the rule $(b^5, c^5)$ allows us to merge nodes $b$ and $c$ to form the new node labeled $b$. We thus get the nondeterministic graph $G_2$:

|  | $e$ | $b$ | $bc$ | $cb$ |
|---|---|---|---|---|
| $b$ | $b$ | $b, cb$ | $b$ | $cb$ |
| $c$ | $b$ | $b, bc$ | $bc$ | $b$ |

Next, we merge $b$ and $cb$, getting $G_3$:

|  | $e$ | $b$ | $bc$ |
|---|---|---|---|
| $b$ | $b$ | $b$ | $b$ |
| $c$ | $b$ | $b, bc$ | $bc$ |

Then we merge $b$ and $bc$, getting $G_4$:

|  | $e$ | $b$ |
|---|---|---|
| $b$ | $b$ | $b$ |
| $c$ | $b$ | $b$ |

In $G_4$, $e$ and $b$ are not eligible for merging, so $p = 4$ and the algorithm is concluded. Since the generator $c$ is not mentioned in the table except as a row header (it is congruent to the generator $b$), it can be eliminated from the list of generators. Thus $M(P)$ is isomorphic to the almost trivial monoid $M(P')$ where $P' = (\{b\}, \{(bb, b)\})$.

The next example illustrates that $n_0$ may be arbitrarily large relative to the lengths of the left and right sides of the rules and also relative to the size of $M(P)$:

**Example 2:** $P = (\Sigma, E)$, where, for any fixed integer $q \geq 3$, $\Sigma = \{0, b_1, \ldots, b_q\}$ and

$$E = \{(b_i, b_1b_2)|\text{ all } i\} \cup \{(b_ib_{i+1}, b_ib_{i+1}b_{i+2})|1 \leq i \leq q-2\}$$
$$\cup\{(b_{q-1}b_q, 0)\} \cup \{(b_i0, 0), (0b_i, 0), (00, 0)|\text{ all } i\}$$

Note that in $M(P)$, all nonnull words are congruent to 0, but the derivation of 0 from any word of length 1 requires a word of length $q$. Since longer words are not needed, $n_0 = q$. This in spite of the fact that $M(P)$ has only two elements, $[e]$ and $[0]$, and no rule involves a word longer than 3. In Section 3 more general things will be said about the size of $n_0$ relative to $P$ and relative to $M(P)$.

This section closes with the observation that the Knuth-Bendix completion procedure [3] can also be used as a semi-algorithm for the problem of whether a given finitely presented monoid is finite. To see this, assume the monoid is finite and consider the function $g$ such that, for every $x \in \Sigma^*$, $g(x) =$ the smallest word $w$ (in the sense of the $<$ relation defined in this section) such that $w \equiv x$. Where $s$ is the order of the monoid then $|g(x)| \leq s - 1$ for every $x$, by Theorem 1.1. If $R'$ is the set of rules

$$\{(x, g(x)) \mid x \in \Sigma^*, |x| \leq s, x \neq g(x)\}$$

then the Thue system $T = (\Sigma, R')$ is a presentation for the monoid and is confluent. Thus every finite monoid has a presentation that is a confluent Thue system. (In [2] it is shown further how to simplify the Thue system to a unique minimal equivalent confluent Thue system.)

Hence, if the Knuth-Bendix completion procedure is applied to a presentation of a monoid that happens to be finite (using the length-lexicographic ordering relation), it will terminate with an equivalent presentation that is confluent. However, it will also terminate in certain other cases where the monoid is infinite, again with an equivalent confluent system (with finitely many rules). Fortunately it is easily decided, for any such Thue system $T$, whether $M(T)$ is finite. If so the multiplication table is easily constructed, without need for the procedure of this section. (Since the set $I$ of irreducible strings of a confluent system is regular, whether or not $I$ is finite is easily decided. And $M(T)$ is finite if and only if $I$ is finite. Indeed, the elements of $M(P)$ can be represented by the words of $I$, as is proved in [2].)

So the Knuth-Bendix completion procedure is a competitor to the procedure of this section. Its advantage is that it can tell us in many interesting instances that the monoid is infinite.

**3. Complexity considerations.** To discuss algorithmic complexity, we must begin with a precise statement of the problem. Actually, there are several ways we could formulate our problem, the most prominent of which is

> (P1) Given a presentation and given that the monoid so presented is finite, what is the multiplication table of the monoid?

With this formulation the procedure of Section 2 is an algorithm. However, if we omit the second "given" phrase, we get a problem that has no algorithm:

> (P2) Given a presentation, what is the multiplication table of the monoid presented?

The procedure of Section 2 is not an algorithm for (P2) because it will not terminate unless the monoid is finite. (It would appear that (P2) has no algorithm, however we modify our concept of "multiplication table.")

This observation will help establish some negative complexity results about the problem (P1). The reasoning will be based on the fact that any procedure applied to a presentation, given that the monoid so presented is finite, can also be applied to any presentation whether or not its monoid turns out to be finite. A familiar example of this reasoning will be given in the proof of the next theorem.

**Conjecture I.** The problem (P1) is not in any recursive time complexity class. That is to say, for every recursive function $f$ and algorithm $A$ for (P1), there exists a presentation $P$ with $M(P)$ finite, such that the computation of $A$ with input $P$ takes more than $f(|P|)$ time units. ($|P|$ is the length of the written expression for the presentation $P$.)

As a credibility argument for this conjecture, a weaker result is now proved.

**Definition.** An *honest* algorithm for (P1) is one that yields, on input $P$, the multiplication table for $M(P)$ in the event that $M(P)$ is finite, but gives no answer at all (either by failing to halt, or by halting with no multiplication table output) in the event that $M(P)$ is infinite. A *partially dishonest* algorithm for (P1) is one that yields the multiplication table for $M(P)$ when $M(P)$ is finite, but, for at least one $P$ with $M(P)$ infinite, yields some (finite) multiplication table, which of course is spurious.

A partially dishonest algorithm is one that is perfectly accurate within the limits of its guarantee, but may give misleading information if as users we go beyond the guarantee by applying it to a presentation $P$ without knowing beforehand that $M(P)$ is finite. In this situation, if the algorithm yields a multiplication table of a finite monoid $M$, we shall be able to conclude only the following: Either $M(P) = M$ or $M(P)$ is infinite.

It might be thought that this uncertainty could be resolved by determining whether or not all the rules of $P$ are valid in $M$. If at least one of them is not then from the guarantee we can conclude that $M(P)$ must be infinite. However, if all the rules of $P$ are valid in $M$ we can conclude only that $M$ is a homomorphic image of $M(P)$, which is possibly infinite. Therefore, when we apply a partially dishonest algorithm for Problem (P1) to a presentation $P$ without having established beforehand that $M(P)$ is finite, we cannot be sure that any multiplication table that is constructed is that of $M(P)$.

Note that the procedure of Section 2 is an honest algorithm for problem (P1).

**Theorem 3.1.** Regarding honest algorithms only, the problem (P1) is not in any recursive time complexity class.

*Proof:* Assume there is a recursive function $f$ representing the complexity of some honest algorithm $A$ that solves (P1). This means that from any presentation $P$ of a monoid given to be finite, $A$ constructs the multiplication table for $M(P)$ in at most $f(|P|)$ units of time. Our theorem is proved by proving in the next paragraph that the existence of such a function $f$ would imply the decidability of the following problem:

(P3) Given a presentation, is the monoid so presented finite?

This problem is known to be undecidable (see [1], pp. 157–160).

We assume the existence of the function $f$. Let $P$ be any presentation. Apply $A$ as a procedure to $P$ and stop it, if it has not already stopped, after $f(|P|) + 1$ time units. If this run lasts $f(|P|) + 1$ time units then $M(P)$ is infinite. On the other hand, since $A$ is honest, if $A$ stops of its own accord before $f(|P|) + 1$ time units, then $M(P)$ is finite if a multiplication table is the output, and is infinite if no such table is the output.$\square$

Another important complexity question remains, which is to relate the time of computation not to the input size but to the output size, which we designate as $s(P)$. We assume that $s(P)$ is a positive integer for every $P$ for which $M(P)$ is finite, but for the purposes of this paper we need not define $s(P)$ precisely. Let $s_0$ be the size of the multiplication table of the trivial monoid, i.e., the monoid whose only element is $e$. We assume that the size of the multiplication table of any other monoid exceeds $s_0$.

**Conjecture II.** There do not exist an algorithm for problem (P1) and a recursive function $f$ such that, for every presentation $P$, the computation time of the algorithm is bounded by $f(s(P))$. That is, for every such algorithm and $f$ there is a $P$ where $M(P)$ is finite but where the computation time exceeds $f(s(P))$.

The question is a significant one. No algorithm for a problem can compute in less time than it takes to write the answer; furthermore, it seems fair to allow it to have additional time as a function of the size of the useful output. More generally, it is interesting to investigate the relation of time of computation to output size as an alternative complexity measure. My argument for Conjecture II begins with two theorems.

**Definition:** $n_0(P)$ = the integral value of $n_0$ when the procedure of Section 2 is applied to the presentation $P$. Where $M(P)$ is infinite we write

$$s(P) = n_0(P) = \infty > i$$

for every positive integer $i$. Also $f(\infty) = \infty$ for any function $f$ for which $\lim_{h\to\infty} f(h) = \infty$.

**Theorem 3.2.** It is undecidable for a given positive integer $q$ and presentation $P$ (including those for which $M(P)$ is infinite) (1) whether $s(P) < q$, and (2) whether $|M(P)| < q$.

*Proof:* (1) The following problem is known to be undecidable (see [1], pp. 157–159): given $P$, is $M(P)$ the trivial monoid? But this problem is reducible to problem (1) as follows: Determine whether $s(P) < s_0 + 1$. If so then $M(P)$ is trivial, if not not.

(2) Similar to the proof for (1), but using the fact that $|M(P)| < 2$ if and only if $M(P)$ is trivial.□

**Theorem 3.3.** There is no recursive function $f$ such that $n_0(P) \leq f(s(P))$ for all $P$ such that $M(P)$ is finite.

*Proof:* Assume $f$ exists and, without loss of generality, assume $f(i+1) > f(i)$ for all $i$. With such a recursive function $f$ the undecidable problem of Theorem 3.2(1) could be decided as follows: For the given $P$, begin the procedure of Section 2 (without knowing whether $M(P)$ is finite or infinite). Recall that this procedure has a variable $n$ that runs through the positive integral values of $n$ until $n = n_0$; that $n_0$ = the smallest $n$ with the property that, for some $k \leq n$, $D(k, n)$ holds; and that this property can be decided effectively for each $n$.

In the present application, keep trying successive values of $n$ until one of two possible things happen:

Case I: $n = n_0$. Complete the procedure of Section 2, which will enable a decision as to whether $s(P) < q$.

Case II: $n = f(q)$. Then, whether or not $M(P)$ is finite, (1) $n_0(P) \geq f(q)$. Whether or not $M(P)$ is finite, we have (2) $n_0(P) \leq f(s(P))$: if $M(P)$ is finite (2) follows from the definition of $f$; if infinite, by our convention about $\infty$. From (1) and (2) we get $f(s(P)) \geq f(q)$. Since $f$ is monotonic increasing, $s(P) \geq q$.□

The proof of Theorem 3.3 would be a proof of Conjecture II if the algorithm of Section 2 were the only algorithm for (P1). What prompts me to put forth this conjecture is my feeling that, whatever algorithm might be discovered for (P1), the value of $n_0$ in each application will play an important rôle in determining the computation time.

The undecidability of the word problem for monoids is well known (see, e.g., [1], pp. 57ff). Thus there is no effective way of testing whether $x \equiv y$ in $M(P)$, for any given words $x$ and $y$ and interpretation $P$. However, there is an effective way of testing whether $x \equiv_i y$. Section 2 demonstrates that, given $P$, once we know the value of $n_0$ and are prepared to test pairs of words for $\equiv_{n_0}$ we have both the knowledge that $M(P)$ is finite and are able to construct its multiplication table. And it would appear that (1) before we know the value of $n_0$ we do not even know whether $M(P)$ is finite, and (2) only by testing words for $\equiv_{n_0}$ can we construct the multiplication table. It therefore is credible that the value of $n_0(P)$ determines in some crucial way the necessary computation time to obtain both the knowledge that $M(P)$ is finite and its multiplication table, whatever algorithm is used.

An interesting related question is, given $P$ and words $w_1$ and $w_2$, where $w_1 \equiv w_2$, what is the smallest value of $i$ such that $w_1 \equiv_i w_2$? The following theorem deals with this question, although it does not seem to add credibility to either of the two conjectures of this section.

**Theorem 3.4.** There is no recursive function $f$ such that, for any $P$ where $M(P)$ is finite and for any words $w_1$ and $w_2$,

$$w_1 \equiv w_2 \text{ if and only if } w_1 \equiv_{f(P,|w_1|,|w_2|)} w_2$$

(Note that this theorem would be easy to prove were it not for the phrase, "where $M(P)$ is finite." In that case we could easily show that the existence of such a recursive function $f$ implies the decidability of the word problem for finitely presented monoids.)

*Proof:* Assume that such a function $f$ exists and, without loss of generality, that $f(P, i, j) < f(P, i, j + 1)$ for all $P, i, j$.

**Lemma.** The existence of such a function $f$ implies that the following problem is decidable: Given positive integer $r$ and presentation P (including presentations for which $M(P)$ is infinite), is every word of length $r$ congruent in $M(P)$ to a shorter word?

*Proof of the Lemma:* Using the function $f$ we can always compute a tentative answer to this question, disregarding the possibility that $M(P)$ may be infinite. This tentative answer will be valid if $M(P)$ is finite but may not be valid if $M(P)$ is infinite.

Suppose first that the tentative answer is "yes." We can enumerate all loop-free derivations from all words of length $r$ in which no word has length exceeding $f(P, r, r-1)$. If we find that there is such a derivation of a shorter word from every word of length $r$ then we know that the correct answer is "yes." On the other hand, if from some word of length $r$ no such derivation yields a shorter word then we know that the tentative "yes" answer was not valid and so $M(P)$ is infinite. But $M(P)$ infinite implies that there is a word of every length not congruent to a shorter word; thus the correct answer is "no."

If the tentative answer is "no" then "no" is the correct answer. For if $M(P)$ is finite then the tentative answer is correct. And if $M(P)$ is infinite then it has a word of *every* length that is not congruent to a word of shorter length and, a fortiori, such a word of length $r$.□

The proof of Theorem 3.4 is completed by using the function $f$ to reduce the problem of the Theorem 3.2(2) (given $P$ and $q$, is $|M(P)| < q$?) to the problem of the Lemma as follows: First use the function $f$ to determine whether or not every word of length $q-1$ is congruent to a shorter word. If so then $M(P)$ is finite; carry through the procedure of Section 2, finding $|M(P)|$ exactly, thus correctly answering the question, is $|M(P)| < q$?

If not, i.e., if there is a word $w$ such that $|w| = q-1$ and $w$ is not congruent to a shorter word, then $|M(P)| \geq q$, by Theorem 1.1.□

Some readers of this section may be disappointed not to see any concrete complexity. In fact, the results of this section indicate that the problem of the paper is beyond concrete complexity. What puts the algorithm of Section 2 beyond practical computation is that the circumstances of its use generally preclude the main advantage of algorithmic computation, namely, the guarantee of a termination and an answer in all instances. In order to have this advantage, users would have to know for some reason that their presentations were those of finite monoids. If they had that information, then probably there would be other available information that could serve to sharpen the procedure to one which could compute within predictable resource limits. It would seem that the theoreticians wanting to extend the results of this paper to complexity-measurable algorithms must be willing to restrict in some way the class of instances to which their algorithms may apply. In other words, they must confine themselves to proper subproblems of the problem (P1).

## References

[1] R.V. Book and F. Otto, *String-rewriting systems,* Springer-Verlag, 1993.

[2] D. Kapur and P. Narendran, The Knuth-Bendix completion procedure and Thue systems, *SIAM J. Computing,* Vol. 14 (1985), pp. 1052–1072.

[3] D.E. Knuth and P.B. Bendix, Simple word problems in universal algebras, in *Computational problems in abstract algebras* (J. Leech, ed.), Pergammon Press, 1970, pp. 263–297.

[4] N.S. Mendelsohn, An algorithmic solution for a word problem in group theory, *Canadian J. Math.,* Vol. 16 (1964), pp 509–516; Correction, Vol. 17 (1965), p. 505.

[5] C.C. Sims, *Computation with finitely presented groups,* Cambridge Univ. Press, 1994.

[6] J.A. Todd and H.S.M. Coxeter, A practical method for enumerating cosets of a finite abstract group, *Proc. Edinburgh Math. Soc.,* Vol. 5 (1936), pp. 26–34.