MIT/LCS/TM-67

COMPUTATIONAL COMPLEXITY OF THE WORD PROBLEM FOR COMMUTATIVE SEMIGROUPS

Edward W. Cardoza

October 1975

TM-67

COMPUTATIONAL COMPLEXITY OF THE WORD PROBLEM FOR COMMUTATIVE SEMIGROUPS

by

Edward Wayne Cardoza

October 1975

This research was supported by the National Science Foundation under research grant GJ-43634X.

COMPUTATIONAL COMPLEXITY OF THE WORD PROBLEM FOR COMMUTATIVE SEMIGROUPS

by

Edward Wayne Cardoza

Submitted to the Department of Electrical Engineering and Computer Science on August 15, 1975 in partial fulfillment of the requirements for the Degree of Master of Science.

ABSTRACT

We analyze the computational complexity of some decision problems for commutative semigroups in terms of time and space on a Turing machine.

The main result we present is that any decision procedure for the word problem for commutative semigroups requires storage space at least proportional to n/logn on a multitape Turing machine. This implies that the word problem is polynomial space hard (and in particular that it is at least NP-hard).

We comment on the close relation of commutative semigroups to vector addition systems and Petri nets.

We also show that the lower bound of space n/logn can be extended to certain other natural algorithmic problems for commutative semigroups. Finally we show that for several other algorithmic problems for commutative semigroups there exist polynomial time algorithms.

THESIS SUPERVISOR: Albert R. Meyer
TITLE: Associate Professor of Computer Science and Engineering

Acknowledgements

I would like to thank my thesis supervisor, Albert R. Meyer, for suggesting the topic and for his suggestions during the course of this thesis.

I would also like to thank Gloria Marshall for her assistance in translating some of the papers from Russian.

This research was supported by the National Science Foundation under research grant GJ-43634X.

Table of Contents

Introduction

We consider in this thesis the computational complexity of an effectively decidable problem of algebra: the word problem for finitely presented commutative semigroups. The main result we present is that any decision procedure for this problem requires space at least proportional to n/log n on a multitape Turing machine.

In section one we define the problem and derive the lower bound.

In section two we survey the known algorithmic solutions to this problem and sketch the close relation of the word problem for commutative semigroups to the reachability problem for vector addition systems and Petri nets [25], [17] (a problem of current interest in theories about parallel processing).

In section three we present an algorithm based upon solutions by Biryukov [6] and Rackoff [39].

In section four we extend the lower bound of space n/log n to certain other natural algorithmic problems for commutative semigroups. Finally we show that for several algorithmic problems for commutative semigroups there exist polynomial time algorithms.

Section 1 - A Lower Bound on the Word Problem for Commutative Semigroups

Definition of Problem

Recall that a semigroup $\langle S, \cdot \rangle$ is a nonempty set S with a binary operation $\cdot$ which is associative. We shall say that a set $G \subseteq S$ is a generating set of elements for the semigroup if every element $s \in S$ can be written as a product of elements in G

$$\text{i.e. } s = a_{i_1} \cdot a_{i_2} \cdot \ldots \cdot a_{i_k} \text{ where } a_{i_j} \in G.$$

It is evident that two different products or "words" can represent the same element in the semigroup. One can then consider the congruence relation R over the set of all words of G such that the relation R holds between two words of G iff they represent the same element of the semigroup.

One common method of presenting a semigroup is in terms of a set G of generators and a set of "defining relations" for these generators, i.e. a set of equations of the form $\{U_i = V_i\}$, where U_i and V_i are words over G. From these equations we derive the relation R for the semigroup as follows: Let X and Y be words over G. We shall say that $X \Rightarrow Y$ iff there exists words X' and X" (possibly empty) and some equation $U_j = V_j$ such that

$$X = X'U_jX'' \text{ and } Y = X'V_jX''$$

$$\text{or } X = X'V_jX'' \text{ and } Y = X'U_jX''.$$

We shall say that X is in the relation R to Y iff $X \overset{*}{\Rightarrow} Y$, i.e. iff there exists a sequence of words $X_1, X_2, \ldots, X_n$, $n \geq 1$, where $X = X_1$ and $X_n = Y$, such that for each i, $1 \leq i < n$, $X_i \Rightarrow X_{i+1}$.

The word problem for finitely presented semigroups is to determine given an arbitrary finite set of generators, a finite set of defining relations, and two words, whether these words are in the relation R for that presentation. In 1947 Emil Post [37] and also A. Markov [34] showed that this problem was recursively unsolvable; in fact they showed a stronger result, that is, they proved that there was a fixed presentation (fixed set of generators and defining relations) for which the word problem was recursively unsolvable. And in fact G.S. Tseiten [51] later showed that the following fixed presentation had a recursively unsolvable word problem:

generators - {a,b,c,d,e}

relations - ac = ca ad = da bc = cb bd = db

eca = ae edb = be abac = abace

However although the general problem is unsolvable, this does not mean that we cannot solve the problem for special cases. In fact consider commutative semigroups; i.e. semigroups in which the generators obey the relation ab = ba for all generators a,b. It is known that the word problem for finitely presented commutative semigroups is recursively solvable [33],[8],[40],[38],[6].

Remark: We can if we wish view word problems for semigroups simply as a derivation problem for a special type of grammar or rewriting system, that is, one in which the rewriting rules are bidirectional or symmetric ("$\leftrightarrow$"), that is, if the grammar includes rewriting rule $\alpha \rightarrow \beta$, it also includes $\beta \rightarrow \alpha$. A common terminology that is used is to describe these symmetric rewriting systems as Thue

systems and to describe the more general nonsymmetric systems as semiThue systems (Post[37]). We shall say that a semiThue system is commutative if it includes the rewriting rules ab → ba; ba → ab for every pair of letters a,b.

Model of Computation

We shall measure the computational complexity of a problem in terms of the time or space required by a Turing machine to solve the problem. The particular variant of the Turing machine model that we shall use shall be the multitape Input-Output Turing machine (IOTM) model defined in Stockmeyer [44].

We informally describe the model as follows. An IOTM M has a finite state control and k + 2 tapes (k>0): an input tape, k worktapes, and an output tape. The input x is placed between the left and right endmarker '$' on the input tape. There are single heads scanning each tape; the input head is 2-way, read-only, the worktape heads are 2-way read/write, and the output head is right-moving, write-only. Depending on the current state of the finite state control and the symbols being scanned by the input and worktape heads, M may in one step change state, print new symbols where the heads are located on the worktapes and on the output tape (but not the input tape) and shift its heads to adjacent locations. We assume the output head shifts right one when and only when it has just printed a symbol on the output tape. Note that since the input head is read-only we are not allowed to write new symbols on the input tape.

We shall say that an IOTM M accepts input x iff given input x M eventually halts after some number of steps in unique accepting state q_a. We say that an IOTM M computes the function f(x) iff M given input x eventually halts in state q_a with f(x) on the output tape. We thus may think of IOTM's either as devices for language recognition or as devices for function computation.

The storage space S(x) used by an IOTM on input x is defined to be the number of different worktape cells visited by the heads of M during the course of its computation on input x.

We define $S(n) = \max_{|x|=n} S(x)$, where $|x|$ denotes the length of x. Note that with the IOTM model it makes sense to talk about a set being accepted or a function being computed in storage space S(n), where S(n) grows more slowly than linearly or more slowly than the length of the output f(x), because tape squares visited by the input tape head and output tape head are not counted as part of the space required by the IOTM.

Complexity of the Word Problem

By the complexity of the word problem for finitely presented commutative semigroups we speak of the complexity of an acceptance or language recognition problem; that is we will denote word problems by strings over some formally defined alphabet and will accept the string iff the word problem which the string denotes is answered affirmatively. We remark now however that there are really three different word problems which we can consider. The first one,

which we denote the fixed word equivalence problem, is the question whether an arbitrary word Y is equivalent to a fixed word X_0 in a fixed presentation P_0 of a commutative semigroup. The second one, which we term the word problem for fixed commutative semigroups, is the question whether two arbitrary words X and Y are equivalent in a fixed presentation P_0. The third one, which we denote the uniform word problem, is the question whether two arbitrary words X and Y are equivalent in an arbitrary presentation P.

For problems one and two we define the following notation.

Let Σ be the set of generators in presentation P_0.

Let Σ^* be the set of all words (strings) over the alphabet Σ;

$$\text{i.e. } \Sigma^* = \{ x_1 \ldots x_n \mid x_i \in \Sigma,\ 1 \le i \le n,\ n \ge 0 \}.$$

An instance of the fixed word equivalence problem will have the form Y, where $Y \in \Sigma^*$.

An instance of the word problem for a fixed commutative semigroup will have the form $X \not\in Y$, where $X, Y \in \Sigma^*$, $\not\in \notin \Sigma$, and $\not\in$ is a special delimiter symbol.

For the uniform word problem however there is no bound on the size or cardinality of the set of generators (since the presentation P varies at each instant). Thus for this problem we shall formally represent a generator by a word in $\{0,1,2\}^*$ of the form $2 \cdot x$, where $x \in \{0,1\}^*$. A word X over the generators is represented in $\{0,1,2\}^*$ by a concatenation of words representing generators. A relation R will be represented by X 3 Y, where $X, Y \in \{0,1,2\}^*$ are words over generators. An instance of the uniform word problem will be a word

in $\{0,1,2,3,4\}^*$ of the form

$$X\ 4\ Y\ 4\ g_1\ g_2 \ldots g_m\ 4\ R_1\ 4\ R_2\ 4 \ldots 4\ R_n$$

where $g_i \in 2\cdot\{0,1\}^*$, $1 \le i \le m$, represent generators, X and Y are words in $\{g_1, \ldots, g_m\}^*$, and $R_j \in \{0,1,2,3\}^*$, $1 \le j \le n$, is a word representing a defining relation for the generators g_i.

Let U.W. be the set of words $\subseteq \{0,1,2,3,4\}^*$ representing instances of the uniform word problem for finitely presented commutative semigroups which are given an affirmative answer.

These (or similar) conventions are necessary, since a fixed Turing machine can recognize words only over a finite alphabet.

By results of Biryukov [6], Taiclin [48], Ginsburg and Spanier [16], and Fischer, Meyer, and Rosenberg [13] one can show that either the fixed word equivalence problem or the word problem for fixed commutative semigroups can be solved in real time on a Turing machine.[1.] (We will discuss these results in more detail in section two and three.) We thus see that the only problem which has a nontrivial lower bound on its complexity is the uniform problem. We now address ourselves to this problem.

Lower Bound

The outline of our lower bound proof is based on the efficient reducibility technique employed in Meyer [36] and Stockmeyer [44] to

---

1. A Turing machine M is termed real time if M moves its input head right at each step and as soon as it finishes reading the input (i.e. in n steps) it halts and accepts or rejects the input.

prove lower bounds, principally for classes of logical theories. Let $A \subseteq \Sigma_1^*$ and $B \subseteq \Sigma_2^*$ for some finite alphabets Σ_1, Σ_2. Roughly, a set A is efficiently reducible to a set B iff there is an "efficiently computable" function f such that a question of the form "Is x in A?" has the same answer as the question "Is f(x) in B?". The particular notion of efficient reducibility that we use is log-space reducibility. We say that $A \underset{\log}{\leq} B$ via f iff f is a function $f: \Sigma_1^* \to \Sigma_2^*$ such that $x \in A$ iff $f(x) \in B$ for all $x \in \Sigma_1^*$ and f is computable using storage space $S(n) = \log n$ on a Turing machine. We say that a function $f: \Sigma_1^* \to \Sigma_2^*$ is length L(n) bounded iff $|f(x)| \leq L(n)$ for all x such that $|x| = n$, where $|x|$ denotes the length of the word x (Stockmeyer [44]).

Let SPACE(S(n)) denote the collection of all sets $A \subseteq \Sigma_1^*$ which can be recognized in storage space S(n) on a (deterministic) Turing machine.

Lemma: Suppose for all sets $A \in \text{SPACE}(n)$, that $A \underset{\log}{\leq} B$ by a length $c_A n \log n$ bounded f, where c_A is a constant depending on A. Then B requires space at least proportional to n/log n on a Turing machine.

Proof: By the space hierarchy theorem (Theorem 10.9 of Hopcroft and Ullman [23]) find a storage bound $S_1(n) \geq \log n$ and a set A such that $A \in \text{SPACE}(n)$, but $A \notin \text{SPACE}(S_1(n))$. By assumption $A \underset{\log}{\leq} B$ by a length cnlogn bounded f, for some constant c. We claim that $S_1(\lceil n/c\log n \rceil)$ is a lower bound on the space complexity of B.

For suppose $B \in SPACE(S_1(\lceil n/c\log n \rceil))$. Then it is easy to see that $A \in SPACE(S_1(n) + \log n)$ (cf. Lemma 3.6 of Stockmeyer [44]). Therefore $A \in SPACE(2 \cdot S_1(n)) = SPACE(S_1(n))$. Contradiction.

Hence $B \notin SPACE(S_1(\lceil n/c\log n \rceil))$. Note that $B \notin SPACE(S_2(n))$ for any function S_2 such that $S_2(n) = o(n/\log n)$, since by the space hierarchy theorem, we can choose any tape constructable $S_1(n) \leq n$ such that $\lim_{n\to\infty} \inf S_1(n) / n = 0$.

Theorem 1: Let $A \subseteq \Sigma^*$ be any set of words (over a finite alphabet Σ) which can be recognized in storage space $S_1(n) = n$ on a (deterministic) Turing machine. Then $A \leq_{\log}$ U.W. via some function f such that $|f(x)| \leq c_A |x| \log_2 |x|$ for some constant $c_A > 0$ and all $x \in \Sigma^*$, $(|x| > 0)$.

Corollary: The uniform word problem for finitely presented commutative semigroups requires space at least proportional to $n/\log n$ on a multitape Turing machine.

Proof of Corollary: The corollary follows directly from the Lemma and Theorem 1.

For purposes of the proof of the Theorem we consider counter (or register) machines [23]. A counter machine consists of a finite state control and some number k ($k \geq 1$) of counters. A counter machine may store a number in each of its counters and may test whether that number is zero or nonzero. One counter is designated an input counter, in which a nonnegative integer input is initially placed; the other counters initially contain zero. Depending on the state and which counters are zero and nonzero, a counter machine may, in a single step, change state and increment or decrement each of the counters by one.

We assume without loss of generality that the counters need never decrement below zero [13]. Let i be a nonnegative integer. The space S'(i) used by a counter machine on input i is defined to be the maximum value stored in any of the counters during the course of a computation.

let $\Sigma = \{s_1, \ldots, s_k\}$ be an arbitrary alphabet.

Let $w = s_{i_0} s_{i_1} \cdots s_{i_t} \in \Sigma^*$.

Let $\mu(w) = \sum_{j=0}^{t} i_j k^j$, i.e. $\mu(w)$ is the number which w denotes in k-adic notation.

This mapping μ establishes a one-one correspondence between strings over Σ and nonegative integers (with λ, the empty string, corresponding to zero).

A counter machine will be said to use space S, where S is a function from nonnegative integers to nonnegative integers, if for all n, $S(n) \geq$ the space used on input i for any integer $i \geq 0$ such that $|\mu^{-1}(i)| = n$; that is, $S(n) \geq \max_{|\mu^{-1}(i)|=n} S'(i)$.

For $A \subseteq \Sigma^*$ define $\mu(A) = \bigcup_{w \in A} \mu(w)$.

Some minor modifications to a theorem in Fischer, Meyer, and Rosenberg [13] yield the following result: Let $S(n) \geq n$. A set A of words is acceptable in space S(n) on a Turing machine iff the set $\mu(A)$ is acceptable in space $k^{S(n)}$ on a counter machine, for some constant $k > 1$. In particular, the sets A acceptable in space $S(n) = n$ on a Turing machine correspond precisely to the sets $\mu(A)$ acceptable in space $S(n) = k^n$ on a counter machine.

Thus to prove Theorem one it suffices to prove

Lemma: Let $A \subseteq \{s_1, s_2\}^*$ be a language. If $\mu(A)$ is recognizable in space $S(n) = k^n$ on a counter machine for some constant k, then $A \leq_{\log}$ U.W. by a length $c_A n \log n$ bounded f.

Proof of Lemma: Let $x = x_1 \cdots x_n \in \{s_1, s_2\}^*$. Let M be a counter machine which operates in space $S(n) = k^n$ and which accepts $\mu(A)$. For expository purposes we assume that M has only two counters. The construction for a larger number of counters is essentially the same as that given below for M's second counter. Assume $k^n \geq 2^{n+1}$, $n \geq 1$.

To prove the lemma we now describe how to construct from x a word $w_x \in \{0,1,2,3,4\}^*$ which is an instance of the uniform word problem such that $\mu(x)$ is accepted by M iff $w_x \in$ U.W. Moreover the construction of w_x will be shown to require only space proportional to $\log|x|$. Of course we shall describe the instance of the uniform word problem in standard language, leaving implicit the coding of generators, relations, etc. into the alphabet $\{0,1,2,3,4\}$.

We now construct a commutative semigroup presentation which (informally speaking) simulates the transition rules of M.

Let m + 1 be the number of states of the counter machine. The presentation contains generators $q_0, q_1, \ldots, q_m$ (for the states of the counter machine), generators a_i, b_i, $1 \leq i \leq n+2$ (for counter one), generators c_j, d, $1 \leq j \leq n+1$ (for counter two), 4(m+1) auxiliary generators, and a generator Q_a.

Consider the following relations.

Counter 1: (Input Counter)

$$a_1a_1 \leftrightarrow a_2 \qquad b_1b_1 \leftrightarrow b_2$$

$$a_2a_2 \leftrightarrow a_3 \qquad b_2b_2 \leftrightarrow b_3$$

$$\vdots \qquad\qquad \vdots$$

$$a_{n+1}a_{n+1} \leftrightarrow a_{n+2} \qquad b_{n+1}b_{n+1} \leftrightarrow b_{n+2}$$

Counter 2:

$$\underbrace{c_1c_1\cdots c_1}_{k} \leftrightarrow c_2$$

$$\underbrace{c_2c_2\cdots c_2}_{k} \leftrightarrow c_3$$

$$\vdots$$

$$\underbrace{c_nc_n\cdots c_n}_{k} \leftrightarrow c_{n+1}$$

Call these relations <u>counter representation rules</u>.

Consider counter one: a word consisting of $2^{n+1} - t$ a_1's and t b_1's, $0 \le t \le 2^{n+1}$, will be said to represent the number t.

Consider counter two: a word consisting of $k^n - j$ c_1's and j d's, $0 \le j \le k^n$, represents the number j.

Note that $a_{n+2} \overset{*}{\leftrightarrow} \underbrace{a_1\cdots a_1}_{2^{n+1}}$ and $c_{n+1} \overset{*}{\leftrightarrow} \underbrace{c_1\cdots c_1}_{k^n}$ by the counter representation rules.

Consider the state transition function δ of M. That is, if

$\delta(q_i,u_1,u_2) = (q_j,v_1,v_2)$ where $u_i \in \{0,+\}$ and $v_i \in \{+1,0,-1\}$ then when M is in state q_i with counter one in zero-nonzero condition u_1 and counter two in zero-nonzero condition u_2, M changes state to state q_j and performs action v_1 on counter one and action v_2 on counter two.

For each state transition rule of M we include two relations in the presentation. For example, if $\delta(q_i,0,0) = (q_j,+1,+1)$ include the relations

$$a_{n+2}q_i c_{n+1} \leftrightarrow a_{n+2}gc_{n+1}$$

$$a_1 gc_1 \leftrightarrow b_1 q_j d$$

where g is an unique auxiliary generator which appears only in these relations.

If $\delta(q_i,+,+) = (q_j,+1,+1)$ then include the relations

$$b_1 q_i d \leftrightarrow b_1 hd$$

$$a_1 hc_1 \leftrightarrow b_1 q_j d$$

where h is an unique auxiliary generator.

If $\delta(q_i,+,+) = (q_j,-1,-1)$ include the relations

$$b_1 q_i d \leftrightarrow b_1 pd$$

$$b_1 pd \leftrightarrow a_1 q_j c_1$$

where p is an unique auxiliary generator.

In general, for $\delta(q_i,u_1,u_2) = (q_j,v_1,v_2)$ include the relations

$$X_1 q_i Y_1 \leftrightarrow X_1 g_{q_i,u_1,u_2} Y_1$$

$$X_2 g_{q_i,u_1,u_2} Y_2 \leftrightarrow X_3 q_j Y_3$$

where if $u_1 = 0$, then $X_1 = a_{n+2}$
$u_1 = +$, $X_1 = b_1$
$u_2 = 0$, $Y_1 = c_{n+1}$
$u_2 = +$, $Y_1 = d$
if $v_1 = +$, then $X_2 = a_1$, $X_3 = b_1$
$v_1 = -$, $X_2 = b_1$, $X_3 = a_1$
$v_1 = 0$, $X_2 = X_1$, $X_3 = X_1$
if $v_2 = +$, then $Y_2 = c_1$, $Y_3 = d$
$v_2 = -$, $Y_2 = d$, $Y_3 = c_1$
$v_2 = 0$, $Y_2 = Y_1$, $Y_3 = Y_1$

and g_{q_i,u_1,u_2} is an unique auxiliary generator.

We call these relations state rules.

The dyadic integer $x = x_1 \cdots x_n$ will be represented in counter one by the word $I_0 = D(x_1) \cdot D(x_2) \cdots D(x_n)$,

where $D(x_i) = \begin{cases} a_i b_i & \text{for } x_i = s_1 \\ b_i b_i & \text{for } x_i = s_2. \end{cases}$

Let $I_x = I_0 a_2$.

By convention the counter machine starts in state q_0 with counter two initially zero. Assume without loss of generality that the counter machine before accepting empties its counters and halts in an unique accepting state q_a.

We add the additional state rule $a_{n+2} q_a c_{n+1} \leftrightarrow Q_a$, where Q_a is a new symbol.

Proposition 1: Given input $\mu(x)$ the counter machine halts in unique accepting state q_a iff the words $I_x q_0 c_{n+1}$ and Q_a are equal in the semigroup presented above.

Proof: Define an instantaneous description (I.D.) of a k^n space bounded counter machine to be the state the counter machine is in and the number (between 0 and 2^{n+1}) that is in counter one and the number (between 0 and k^n) that is stored in counter two. We shall say for two i.d.'s I,J that J is the next i.d. of I iff instantaneous description J follows in one move from instantaneous description I.

Let #a in X denote the number of occurrences of the generator a in the word X. We shall drop the phrase "in X" if the word X is understood from the context.

Define an i.d. word as follows:

Let $Q' = \{\text{state symbols } q_i\} \cup \{\text{auxiliary generators}\}$.

A word is an i.d. word iff it contains exactly one letter $q' \in Q'$ and $\#a_1 + 2\cdot\#a_2 + \ldots + 2^{n+1}\cdot\#a_{n+2} + \#b_1 + 2\cdot\#b_2 + \ldots + 2^{n+1}\cdot\#b_{n+2} = 2^{n+1}$ and $\#c_1 + k\cdot\#c_2 + \ldots + k^n\cdot\#c_{n+1} + \#d = k^n$ and the only letters the word contains are a member of the above sets (i.e. $Q' \cup \{a_i, b_i, c_i, d\}$).

We shall call two i.d. words X and Y congruent if X and Y contain the same letter $q' \in Q'$ and $\#b_1 + 2\cdot\#b_2 + \ldots + 2^{n+1}\cdot\#b_{n+2}$ in X $= \#b_1 + 2\cdot\#b_2 + \ldots + 2^{n+1}\cdot\#b_{n+2}$ in Y and $\#d$ in X $= \#d$ in Y. Note that if X and Y are i.d. words and $X \overset{*}{\leftrightarrow} Y$ by use of only the commutativity and counter representation relations then X and Y are congruent i.d. words, and conversely. Note also that all the rules, except for the one involving the terminal symbol Q_a, preserve the property of a word being an i.d. word.

Now, with every i.d. J of the counter machine we associate an expanded i.d. word $W_J = qa_1^t b_1^m c_1^r d^p$, where q is the state of the counter machine, m is the number stored in counter one, p is the number stored in counter two, t equals $2^{n+1} - m$, and r equals $k^n - p$.

Observe that $I_x q_0 c_{n+1}$ corresponds to the initial i.d. on input $\mu(x)$.

Let T be the semigroup presentation given above. Note that we can consider T to be a commutative Thue system. Consider the commutative semiThue system S which consists of T's commutative and counter representation rules (both directions "$\leftrightarrow$") and the forward directional state rules ("$\rightarrow$") of T. Let $\overset{}{\Rightarrow}_S$ be the derives relation for the commutative semiThue system S and let $\overset{*}{\Rightarrow}_S$ be its reflexive transitive

completion. Define $\overset{}{\underset{T}{\Rightarrow}}$ and $\overset{*}{\underset{T}{\Rightarrow}}$ similarly for T.

Lemma 1: If J' is the next i.d. of J, then $W_J \overset{*}{\underset{S}{\Rightarrow}} W_{J'}$.

Proof: This should be clear from the construction. For example if J = (q,9,11) and $\delta(q,+,+) = (q',+,+)$

$$W_J = qa_1^{2^{n+1}-9}b_1 9\, 9\, k^n\text{-}11\, d\, 11$$

$$W_J \overset{*}{\underset{S}{\Rightarrow}} a_1^{2^{n+1}-9}b_1 9\, q d\, 11\, c_1 k^n\text{-}11 \qquad \text{by the commutativity relations}$$

$$\overset{}{\underset{S}{\Rightarrow}} a_1^{2^{n+1}-9}b_1 9\, h d\, 11\, c_1 k^n\text{-}11 \qquad \text{state rule}$$

$$\overset{*}{\underset{S}{\Rightarrow}} b_1 9\, a_1^{2^{n+1}-9}h c_1 k^n\text{-}11\, d\, 11 \qquad \text{commutativity}$$

$$\overset{}{\underset{S}{\Rightarrow}} b_1 9\, a_1^{2^{n+1}-10}b_1 q' d c_1 k^n\text{-}12\, d\, 11 \qquad \text{state rule}$$

$$\overset{*}{\underset{S}{\Rightarrow}} q' a_1^{2^{n+1}-10}b_1 10\, c_1 k^n\text{-}12\, d\, 12 \qquad \text{commutativity}$$

We remark that if $\#d = k^n$ or $\#b_1 = 2^{n+1}$ then we cannot carry out this derivation; but then of course M by hypothesis does not increase counter one when it contains 2^{n+1} or counter two when it contains k^n.

Lemma 2: Let X and Y be two i.d. words. Assume $X \overset{*}{\leftrightarrow} Y$ by use of only the counter representation and commutativity relations. Assume $X \overset{}{\underset{S}{\Rightarrow}} Z_1$ by a state rule and $Y \overset{}{\underset{S}{\Rightarrow}} Z_2$ by a state rule. Then the state rule used is the same and $Z_1 \overset{*}{\leftrightarrow} Z_2$ by use of only the counter representation and commutativity relations.

Proof: Since X and Y are congruent i.d. words, X and Y must contain the same state or auxiliary generator symbol q.

Case 1: If q is an auxiliary generator symbol, then the state

rule applicable at both X and Y must have been the same one, since an auxiliary generator symbol appears on the left hand side of only one of the state relations.

A state rule of this case replaces an auxiliary generator symbol by a state symbol and changes c_1 to a d (or vice versa) and b_1 to an a_1 (or vice versa). Hence i.d. words Z_1 and Z_2 have the same q symbol, $\#d$ in Z_1 = $\#d$ in Z_2, and $\#b_1 + \ldots + 2^{n+1} \cdot \#b_{n+2}$ in Z_1 = $\#b_1 + \ldots + 2^{n+1} \cdot \#b_{n+2}$ in Z_2. Hence Z_1 and Z_2 are congruent i.d. words.

Case 2: Let q be a state symbol. Since X and Y are congruent i.d. words, $\#d$ in X = $\#d$ in Y and $\#b_1 + \ldots + 2^{n+1} \cdot \#b_{n+2}$ in X $= \#b_1 + \ldots + 2^{n+1} \cdot \#b_{n+2}$ in Y. By definition of i.d. words the presence of a c_{n+1} and d, (and also a_{n+2} and b_1) is mutually exclusive in an i.d. word. But there is only one state rule applicable for each $q \cdot (c_{n+1}$ or $d) \cdot (a_{n+2}$ or $b_1)$ combination. Hence the state rule applicable at congruent i.d. words X and Y must have been the same one.

If the state rule used was $a_{n+2} q_a c_{n+1} \rightarrow Q_a$, then necessarily $X = Y = a_{n+2} q_a c_{n+1}$ and obviously $Q_a \overset{*}{\leftrightarrow} Q_a$.

Otherwise, the state rule used simply replaced a state symbol by an auxiliary generator symbol. Since the same change is made in X and Y, Z_1 and Z_2 are congruent i.d. words.

From Lemmas 1 and 2 and an induction on the number of state rules used in a derivation follows:

Lemma 3: Given input $\mu(x)$, the counter machine halts in unique

accepting state q_a iff $Iq_0c_{n+1} \overset{*}{\underset{S}{\Rightarrow}} Q_a$ in the commutative semiThue system S.

The commutative semiThue system S has a Church-Rosser-like property:

Lemma 4: If X is an i.d. word and $X \overset{*}{\underset{S}{\Rightarrow}} Y_1$ and $X \overset{*}{\underset{S}{\Rightarrow}} Y_2$ then there exists Z such that $Y_1 \overset{*}{\underset{S}{\Rightarrow}} Z$ and $Y_2 \overset{*}{\underset{S}{\Rightarrow}} Z$.

Proof: Consider the following diagram:

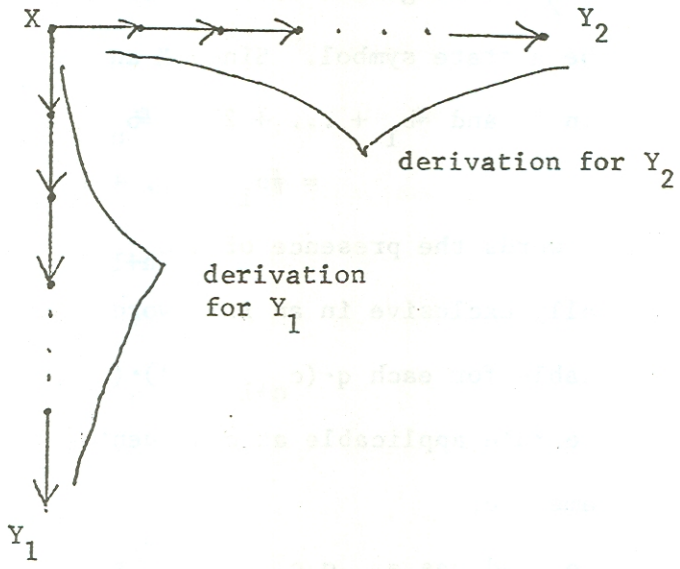

Figure 1

Consider the first time in which a state rule is applied in the derivation for Y_1 and the first time in which a state rule is applied in the derivation for Y_2. (If there are no state rules used in either derivation or only on one then the proof is obvious.)

i.e.:

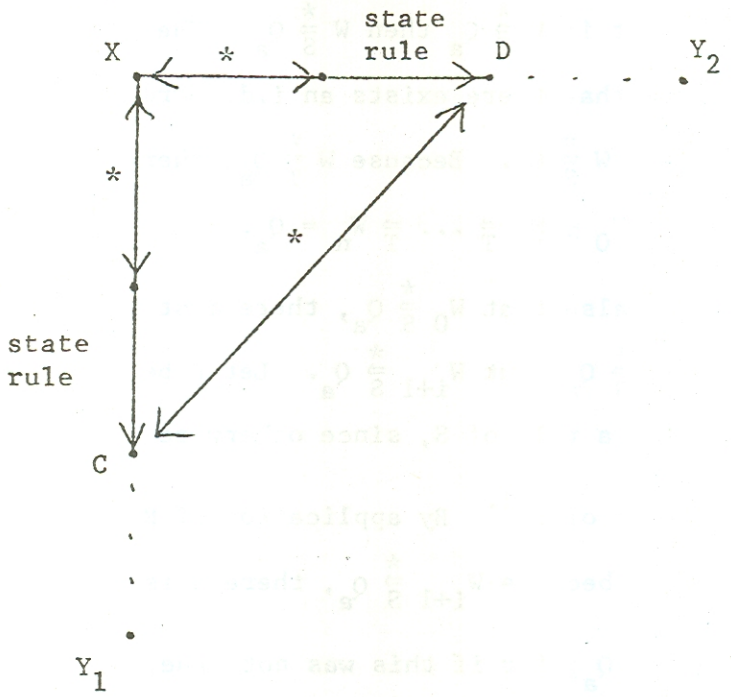

Figure 2

But then according to lemma 2, $C \overset{*}{\leftrightarrow} D$ by use of only the commutativity and the counter representation relations. By continuing in this manner (that is, by induction on the sum of the number of state rules used in the derivation for Y_1 and Y_2) we get that either $Y_1 \overset{*}{\underset{S}{\Rightarrow}} Y_2$ or $Y_2 \overset{*}{\underset{S}{\Rightarrow}} Y_1$ (or both). Hence there exists Z such that $Y_1 \overset{*}{\underset{S}{\Rightarrow}} Z$ and $Y_2 \overset{*}{\underset{S}{\Rightarrow}} Z$.

Lemma 5: For every i.d. word W, $W \overset{*}{\underset{T}{\Rightarrow}} Q_a$ iff $W \overset{*}{\underset{S}{\Rightarrow}} Q_a$.

Proof[1.]: It is obvious that if $W \overset{*}{\underset{S}{\Rightarrow}} Q_a$, then $W \overset{*}{\underset{T}{\Rightarrow}} Q_a$, since every rule of S is also a rule of T.

---

1. Parts of the proof of this lemma are based upon the argument originally used by Post in [37].

We now show that if $W \overset{*}{\Rightarrow}_T Q_a$ then $W \overset{*}{\Rightarrow}_S Q_a$. The proof is by contradiction. We assume that there exists an i.d. word W for which $W \overset{*}{\Rightarrow}_T Q_a$, but it is false that $W \overset{*}{\Rightarrow}_S Q_a$. Because $W \overset{*}{\Rightarrow}_T Q_a$, there must be a derivation

$$W = W_0 \Rightarrow_T W_1 \Rightarrow_T \cdots \Rightarrow_T W_n = Q_a.$$

Because it is false that $W_0 \overset{*}{\Rightarrow}_S Q_a$, there must be an $i < n$ such that it is false that $W_i \overset{*}{\Rightarrow}_S Q_a$, but $W_{i+1} \overset{*}{\Rightarrow}_S Q_a$. Let R be the rule of T used for $W_i \Rightarrow_T W_{i+1}$. R is not a rule of S, since otherwise $W_i \overset{*}{\Rightarrow}_S Q_a$. Thus the inverse R^{-1} is a rule of S.[1] By application of R^{-1} we have that $W_{i+1} \Rightarrow_S W_i$. Further, because $W_{i+1} \overset{*}{\Rightarrow}_S Q_a$, there exists $\overline{W}$ such that such that $W_{i+1} \Rightarrow_S \overline{W} \overset{*}{\Rightarrow}_S Q_a$; for if this was not true, $W_{i+1} = Q_a$; but then $Q_a \Rightarrow_S W_i$, which is impossible since no rule of S is applicable to Q_a.

From $W_{i+1} \Rightarrow_S W_i$ and $W_{i+1} \Rightarrow_S \overline{W}$, we get by lemma 4 that there exists W' such that $W_i \overset{*}{\Rightarrow}_S W'$ and $\overline{W} \overset{*}{\Rightarrow}_S W'$. But $\overline{W} \overset{*}{\Rightarrow}_S Q_a$ also. Hence by another application of lemma 4 we get that there exists W'' such that $W' \overset{*}{\Rightarrow}_S W''$ and $Q_a \overset{*}{\Rightarrow}_S W''$. But then W'' must equal Q_a, since the only W'' such that $Q_a \overset{*}{\Rightarrow}_S W''$ is Q_a. Hence $W_i \overset{*}{\Rightarrow}_S W' \overset{*}{\Rightarrow}_S W'' = Q_a$, contradicting our assumption.

From Lemmas 3 and 5 Proposition 1 follows.

Proposition 2: Let $x \in \{s_1, s_2\}^*$. The presentation constructed above for x is log-space computable by an IOTM.

Proof: We seek to construct an IOTM which given x will print out the

---

1. The inverse of a rule $A \rightarrow B$ is defined to be $B \rightarrow A$.

presentation on the output tape using storage space $S(n) = \log n$ on the worktapes.

Let $x = x_1 \ldots x_n$.

It is clear that the counter representation relations

$$a_1 a_1 \leftrightarrow a_2$$

$$a_2 a_2 \leftrightarrow a_3$$

$$\vdots$$

$$a_{n+1} a_{n+1} \leftrightarrow a_{n+2}$$

may be computed from x using storage space $S(n) = \log n$: a portion of the worktape for recording the subscript number of a generator is sufficient.

Consider the state rules, for example $a_{n+2} q_i c_{n+1} \leftrightarrow a_{n+2} g c_{n+1}$

$$a_1 g c_1 \leftrightarrow b q_j d \quad .$$

There will be a fixed number of tracks on the worktape to record a_{n+2}, c_{n+1}, a_1, c_1, b_1, d's subscript number. The IOTM will have the information of the counter machine's state table within its state table. The IOTM then simply cycles through all state and counter position possibilities. The IOTM will have one track to record where we are in the subscripting of the new generators.

The word I may be constructed in log space, since the construction is $\begin{cases} a_i b_i \text{ if } x_i = s_1 \\ b_i b_i \text{ if } x_i = s_2 \end{cases}$, $1 \le i \le n$

and this can be done in log space. Thus the two words can be printed in log space. It is also clear that the IOTM can print all the

generators in log space (since it kept track of the highest subscript number used). □

Note that the presentation constructed used at most $c_1 n$ generators and $c_1 n$ relations for some constant c_1 (depending on the machine). Each relation was of fixed size. The word I was of length proportional to n. Hence, since each generator may need log n symbols to represent it, we have

Proposition 3: The length of the presentation constructed above is of length less than or equal to cnlogn for some constant c.

This completes the proof of Theorem 1.

An alternative proof of the theorem

A.R. Meyer has observed that it is possible to prove our lower bound of space n/logn directly in terms of linear space-bounded Turing machines. We sketch now this proof.

Let $x = x_1 \ldots x_n$ be an input word in $\{0,1\}^*$ and let M be a linear space-bounded Turing machine with state transition function δ.

We again construct a presentation which (informally speaking) simulates the state transition function of M. The presentation constructed uses generators a_i, b_i, $1 \le i \le n$, and generators q_j^i, $1 \le i \le n$, $0 \le j \le m$, where $m + 1$ is the number of states of the Turing machine.

Let I be the word $q_0^1 f(x_1) f(x_2) \cdots f(x_n)$

$$\text{where } f(x_i) = \begin{cases} a_i & x_i = 0 \\ b_i & x_i = 1, \ 1 \le i \le n. \end{cases}$$

Let $\delta(q_j,y) = (q_k,D,z)$, where $y \in \{0,1\}$, $D \in \{L,R\}$, and $z \in \{0,1\}$, denote that if the Turing machine is in state q_j reading symbol y, then it changes state to state q_k, prints new symbol z, and moves the head in direction D one cell (L denotes left, R denotes right).

For each state transition rule of M we construct n - 1 relations in the presentation.

For example, if $\delta(q_j,0) = (q_k,L,z)$, where z = 0 or 1 then we include for each i, $1<i\leq n$, the relations

$$a_i q_j^i \leftrightarrow a_i q_k^{i-1} \quad \text{,if } z = 0$$
$$a_i q_j^i \leftrightarrow b_i q_k^{i-1} \quad \text{,if } z = 1.$$

If $\delta(q_j,0) = (q_k,R,z)$, where z = 0 or 1, we include for $1\leq i<n$:

$$a_i q_j^i \leftrightarrow a_i q_k^{i+1} \quad \text{, if } z = 0$$
$$a_i q_j^i \leftrightarrow b_i q_k^{i+1} \quad \text{, if } z = 1.$$

If $\delta(q_j,1) = (q_k,L,z)$, then for $1<i\leq n$:

$$b_i q_j^i \leftrightarrow a_i q_k^{i-1} \quad \text{, if } z = 0$$
$$b_i q_j^i \leftrightarrow b_i q_k^{i-1} \quad \text{, if } z = 1.$$

If $\delta(q_j,1) = (q_k,R,z)$, then for $1\leq i<n$:

$$b_i q_j^i \leftrightarrow a_i q_k^{i+1} \quad \text{, if } z = 0$$
$$b_i q_j^i \leftrightarrow b_i q_k^{i+1} \quad \text{, if } z = 1.$$

Assume without loss of generality that M halts in unique accepting state q_a, scanning its leftmost cell, with the tape containing all zeros. Then the two words we test for equality in the semigroup are I and $q_a^1 a_1 a_2 \cdots a_n$.

The remaining details of the proof are analagous to the one

presented above.

Remark on Alternate Notation

It should be noted that the definition used for the size of the word problem used an unary-like notation; that is, one wrote "$\underbrace{a \cdot a \cdot \ldots a}_{k}$" rather than the exponent notation "a^z" where z is a binary (or some other radix) notation for k.

Proposition: Consider uniform word problems for commutative semigroups. One can translate efficiently (that is, in log space) a word problem of size n in the exponent notation to an equivalent one of size at most 5nlogn in the unary-like notation.

Proof: Suppose for example that in the exponent notation we had the relation

$$a*z_{j_1}\, b*z_{k_1}\, c*z_{t_1} = a*z_{j_2}\, b*z_{k_2}\, c*z_{t_2},$$ where a,b,c are generators, z_i (the exponent) is a binary number of length i, and * is a special delimiter symbol.

Let $j = \max(j_1,j_2)$; $k = \max(k_1,k_2)$; $t = \max(t_1,t_2)$.

In the unary-like notation we would include the relations

$$aa = a_1 \qquad bb = a_{j+1} \qquad cc = a_{j+k+1}$$

$$a_1a_1 = a_2 \qquad a_{j+1}a_{j+1} = a_{j+2} \qquad a_{j+k+1}a_{j+k+1} = a_{j+k+2}$$

$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$$

$$a_{j-2}a_{j-2} = a_{j-1} \qquad a_{j+k-2}a_{j+k-2} = a_{j+k-1} \qquad a_{j+k+t-2}a_{j+k+t-2} = a_{j+k+t-1}$$

and the relation

$$\overbrace{\quad}^{j_1 \text{ letters}}\ \overbrace{\quad}^{k_1 \text{ letters}}\ \overbrace{\quad}^{t_1 \text{ letters}} = \overbrace{\quad}^{j_2 \text{ letters}}\ \overbrace{\quad}^{k_2 \text{ letters}}\ \overbrace{\quad}^{t_2 \text{ letters}} \qquad (1)$$

where the letters in (1) are determined by a method analogous to the one used in our lower bound proof, e.g. $a^{*}z_{j_1}$, where $z_{j_1} = x_{j_1} \ldots x_1$ is transformed to $\overbrace{f(x_{j_1}) \ldots f(x_1)}^{j_1 \text{ letters}}$, where

$$f(x_i) = \begin{cases} a & x_i = 1 \text{ and } i = 1 \\ a_{i-1} & x_i = 1 \text{ and } i \neq 1 \\ \lambda \text{ (empty string)} & x_i = 0. \end{cases}$$

Thus instead of a relation in the exponent notation of length $j_1 + k_1 + t_1 + j_2 + k_2 + t_2$ we would have relations of total length less than or equal to $\log(j + k + t)(j_1 + k_1 + t_1 + j_2 + k_2 + t_2)$ $+ 4(j+k+t)\log(j+k+t)$ in the unary notation.

In general, let S be the sum of the lengths of the binary exponents used in all the relations in the exponent notation. Let R be the total length of all the relations in the exponent notation. Then in the corresponding unary-like notation the total length of the relations would be at most $5\log S \cdot R$.

A similar method works for the two input words. Hence we can construct an equivalent word problem of size at most $5n\log n$ in the unary-like notation. □

We remark that we can improve the lower bound for the uniform word problen by a log n factor if we allow presentations in exponent notation. In fact with this notation one can show that the uniform

word problem for commutative semigroups with eight generators requires at least linear space on a Turing machine. (This follows directly by the counter machine proof.) Indeed the number of generators needed appears to be the major distinction between the two notations.

We remark also that if the fixed word equivalence problem or the word problem for fixed commutative semigroups uses exponent notation, then these problems can be solved in linear time on a Turing machine. (These latter results will be discussed in more detail at the end of section three.)

Remark: If one uses a double exponential notation, that is, if one writes $a_1^{2^z}$, where z is a number represented in binary (so that one can express numbers of size 2^{2^n} with a length n formula), then one can show that an exponential space lower bound holds for the uniform problem (in fact for commutative semigroups with only 8 generators). Moreover an analysis of the Emelichev-Hermann solution shows that one also can obtain an exponential space upper bound for the problem with this notation (in fact for arbitrary number of generators). (These latter results will be discussed in more detail in section two.)

Section 2 - Commutative Semigroups and Vector Addition Systems

As mentioned earlier, it is known that the word problem for finitely presented commutative semigroups is solvable; in fact there are at least five different solutions cited in the literature: Malcev [33], Emelichev [8], Tseiten [40], Rabin [38], and Biryukov [6]. We have however at present been unable to obtain two of these solutions, Tseiten's and Rabin's. We now briefly note the remaining three.

1. Malcev showed that finitely presented commutative semigroups are residually finite and hence they have a solvable word problem. [A semigroup S is residually finite iff for any two elements $a,b \in S$, $a \neq b$, there exists a homomorphism h of S onto a finite semigroup such that $h(a) \neq h(b)$.] Basically what this says is the following: we know that we can recursively enumerate all pairs of equivalent words. The residual finiteness property is then used to recursively enumerate pairs of inequivalent words. No bound on complexity follows from this proof.

2. Biryukov has shown that the set of words equivalent to a fixed word in a commutative semigroup is an effectively constructable semilinear set. From this construction the solvability of the word problem follows easily. However the procedure given by Biryukov for constructing this set relies for termination upon the theorem that any sequence of incomparable elements in N^k (partially ordered in the usual way) is finite. One can show that, in terms of the size of the initial vector, the maximum difference between two elements in the sequence, and the number of dimensions, that a lower bound on the possible length of an

arbitrary incomparable sequence is the nonprimitive recursive function Ackermann's function [18]. Thus unless one can give a new proof for termination for Biryukov's procedure, we are not able to derive a good upper bound from it.[1.]

3. We have recently obtained a copy of Emelichev's solution. Emelichev points out that the word problem for commutative semigroups is closely related to a subcase of the membership problem for a polynomial in a finitely generated polynomial ideal. From a paper by Hermann[22] and also of Seidenberg [42] on the latter subject and a few obvious simplifications we can show that an exponential space upper bound holds for the uniform word problem for commutative semigroups. (This solution will be presented in more detail in a forthcoming paper.)

Vector Addition Systems

We discuss now the close relation of the word problem for commutative semigroups to the reachability problem for vector addition systems,[2.] an open problem in the theory of computation of several years standing [25].

Definition: Let N denote the nonnegative integers. Let Z denote the integers. A vector addition system V is a pair $\langle I,P\rangle$ where I, the initial vector, is a vector in N^k, and P, the set of periods, is a finite set of vectors in Z^k.

---

1. I should note that the solvability of the equivalence, inclusion, and many other problems also follows from this construction (ref. section 4).

2. This relation was apparently first observed by Michael Rabin [25].

The reachability set R(V) for the vector addition system V is the set of all vectors of the form $I + p_1 + \ldots + p_n$ such that $\forall i \leq n$ $(n \geq 0)$, $p_i \in P$ and $I + \sum_{j=1}^{i} p_j \geq 0$.

The uniform reachability problem is: given a vector addition system V and vector J, is $J \in R(V)$?

We show that the word problem for commutative semiThue systems is recursively equivalent to the reachability problem.

Consider a commutative semiThue system S. Note that in general a rule of S might have the same letters on both sides, for example $a \rightarrow ab$. We now show:

Lemma: For any commutative semiThue system S over the alphabet $\{a_1, \ldots, a_n\}$ we can find a commutative semiThue system S' over the 2n letters $\{a_1, \ldots, a_n, a_1', \ldots, a_n'\}$ such that

1) $X \overset{*}{\Rightarrow} Y$ in S iff $X \overset{*}{\Rightarrow} Y$ in S' and

2) S' does not have the same letter on both sides in any of its relations.

Proof: For every relation $A \rightarrow B$ in S we form the relation $A \rightarrow B'$ in S', where B' is the result of substituting a_i' for each a_i in B.

In addition we include in S' the relation $a_i \leftrightarrow a_i'$ for each i, $1 \leq i \leq n$.

It should be clear that the S' constructed satisfies conditions 1) and 2).

For a word X over the alphabet $\{a_1, \ldots, a_n\}$ define "$\#a_i$ in X" to be the number of occurrences of the letter a_i in X.

Let S now satisfy condition 2 of the lemma above. With each relation r: $A \to B$ in S we associate the integer vector $p_r = (x_1, \ldots, x_n)$,

$$\text{where } x_i = \begin{cases} -(\#a_i \text{ in } A), & \text{if } a_i \text{ occurs in } A \\ \#a_i \text{ in } B, & \text{if } a_i \text{ occurs in } B \\ 0, & \text{otherwise.} \end{cases}$$

With a word X over the alphabet $\{a_1, \ldots, a_n\}$ we associate the non-negative vector $v_X = (x_1, \ldots, x_n)$, where $x_i = \#a_i$ in X. With a commutative semiThue system S we associate the set of periods P_S, where $P_S = \bigcup_{r \in R} p_r$, R the set of relations in S. Then

Lemma: For all words X, Y and commutative semiThue systems S $X \overset{*}{\Rightarrow} Y$ in S iff $v_Y \in R(< v_X, P_S >)$.

By reversing the above correspondence we can for each vector addition system $V = < I, P >$ construct a commutative semiThue system S_V and a word X_I such that $J \in R(< I, P >)$ iff $X_I \overset{*}{\Rightarrow} X_J$ in S_V.

We thus see that the word problem for commutative semiThue systems is recursively solvable iff the reachability problem for vector addition systems is recursively solvable. Clearly then the word problem for commutative semigroups is simply the subcase in which all the rewriting rules are bidirectional or symmetric.

Remark: An alternate notational device often used instead of vector addition systems is Petri nets. For a definition of this notation see [17], [18], or [26].

Section 3 - Semilinearity and Commutative Semigroups

In this section we shall present an algorithm for the construction of the semilinear set of equivalent words to a given word in a commutative semigroup. The algorithm presented here is based in part upon the solution by Biryukov and in part upon a slightly different solution by Rackoff [39].

Let $I \in N^k$, $P \subseteq N^k$, P finite. Define a linear set $L(I;P)$ to be the set of all vectors of the form $I + n_1P_1 + \ldots + n_mP_m$, $n_i \in N$, $P_i \in P$, $1 \leq i \leq m$. If $P = \emptyset$, let $L(I;P) = \{I\}$. Note that a linear set can be considered as a vector addition system in which the periods are all nonnegative. Define a semilinear set S to be a finite union of linear sets all of the same dimension; i.e. $S = L_1 \cup L_2 \cup \ldots \cup L_t$, where L_i is a linear set. Let $J = \{J_1, \ldots, J_r\}$, where $J_i \in N^k$. Define $L(J;P)$ to be $L(J_1;P) \cup L(J_2;P) \cup \ldots \cup L(J_r;P)$.

Let $A = (a_1, \ldots, a_k)$ and $B = (b_1, \ldots, b_k)$ be vectors in N^k. Define the relation "$\leq$" between two vectors $A,B \in N^k$: $A \leq B$ iff $a_i \leq b_i$ for each i, $1 \leq i \leq k$. Define $A < B$ iff $A \leq B$ and $A \neq B$. We shall say that two vectors $A,B \in N^k$ are incomparable if neither $A \leq B$ holds nor $A \geq B$ holds. The following is a well-known result [17]: Any set of mutually incomparable elements in N^k is finite. Let T be a subset of N^k; define min T to be the set of minimal elements of T (with respect to the relation "$<$"). Note that min T, being a set of incomparable vectors, is always finite.

Note that we can identify words given over k generators in a commutative semigroup with k-tuples of nonnegative integers. We shall identify two words X and Y which correspond to the same k-tuple; we shall say that $X \equiv Y$ for two words X,Y iff X equals Y except for a possible reordering of the letters. Because of this identification between words and k-tuples of nonnegative integers, we shall freely switch between the notation and operations of one and the other whenever it is convenient to do so.

Theorem 1: The set of words equivalent to any given word in a finitely presented commutative semigroup is a semilinear set.[1.]

Nonconstructive Proof: Let C be the given word.

Definition: Let A and B be words. We shall say that B is a unit for A iff AB is equivalent to A in the commutative semigroup.

Let E = min{words equivalent to C}.

Let U = min{words that are units for C}.

Claim: S = L(E;U) is the set of all words equivalent to C.

Proof: 1. It is clear that if $Y \in S$ then Y is equivalent to C.

2. Consider an arbitrary word Y equivalent to C. If Y is a minimal such word then it is included in E. If it is not, then there exists $e_i \in E$ such that $Y \equiv e_i Y'$. Since the quotient[2.] of two equivalent words is a unit (for those words), Y' is a unit. If Y' is a minimal unit for C, then $Y' \in U$. Otherwise there exists $D_i \in U$ such that

---

1. Since every finitely generated commutative semigroup is finitely presented [40], [14], this result in fact holds for any finitely generated commutative semigroup.

2. Z is the quotient of words X and Y iff $X \equiv YZ$.

$Y' \equiv D_i Y''$, and (since the quotient of two units is obviously a unit) Y'' is a unit. Continuing in this manner we see that Y' is expressible by a product of units in U.

Constructive Proof: Let $K(G,M)$ denote the commutative semigroup given by the generators $G = \{a_1, \ldots, a_n\}$ and the defining relations $M = \{A_j = B_j \mid j = 1, \ldots, m\}$.

We construct two sequences $E_0, E_1, \ldots, \ldots$ and $U_0, U_1, \ldots,$ of finite sets of nonempty words. Here the set E_i contains certain words equal in $K(G,M)$ to C, while the set U_i contains some of the units of C. In order to construct the sequences we set

1) $E_0 = \{C\}$; $U_0 = \phi$.

2) Assume we have already constructed E_i, U_i.

Let $\alpha = \max_{1 \le j \le m} \{|A_j|, |B_j|\}$, where for a word W, $|W|$ denotes the length of the word W.

Assume $U_i = \{D_1, \ldots, D_k\}$.

We will let E_i' denote the set consisting of all those words D such that

$$C_\ell D_1^{r_1} \cdots D_k^{r_k} \equiv PA_j = PB_j \text{ in the semigroup} \equiv D,$$

for some $C_\ell \in E_i$, some word P, some integers r_t, $0 \le r_t \le \alpha$, $1 \le t \le k$, and some equation $(A_j = B_j) \in M$.

We let U_i' denote the set of words obtained by adding all possible quotients of pairs of words of $(E_i \cup E_i')$ to U_i. Denote the closure of U_i' with respect to quotients by U_i''. Note that U_i'' is finite (assuming E_i and U_i are).

Define $E_{i+1} = \min(E_i \cup E_i')$ and

$$U_{i+1} = \min U_i''.$$

The algorithm terminates when we find an $i + 1$ such that $E_i = E_{i+1}$ and $U_i = U_{i+1}$.

Proposition 1: The algorithm given above terminates.

Proof: The sets E_i have the following two properties by construction:

Property One: The elements of a set E_i are mutually incomparable.

Property Two: To form E_{i+1} no element of E_i is discarded unless it is replaced by a smaller element; i.e. for every element $e \in E_i - E_{i+1}$ there is a strictly smaller element $e_0 \in E_{i+1}$.

Lemma 1: An element of E_{i+1} is not strictly greater than any element of any E_j, $j \leq i+1$.

Proof: Let $e \in E_{i+1}$; we wish to show that $\forall x \in E_j$, $0 \leq j \leq i+1$, it is false that $x < e$.

Assume the contrary; let j be the largest integer $\leq i+1$ such that there exists $x \in E_j$ such that $x < e$. By property one, $j \leq i$. Then property two would imply the existence of an element $x_0 \in E_{j+1}$ such that $x_0 \leq x$. But then $x_0 < e$, contradicting maximality of j.

Lemma 2: Any element of E_i is greater than or equal to some element of E_j for each $j \geq i$.

Proof: Directly from property two.

Lemma 3: If $E_i \neq E_{i+1}$, then there exists e such that $e \in E_{i+1}$, but $e \notin E_i$.

Proof: Directly from property two.

Lemma 4: $E_{i+1} - E_i = E_{i+1} - \bigcup_{j \leq i} E_j$.

Proof: We need only show that if $e \in E_{i+1} - E_i$, then $e \notin E_j$ for $j < i$.

Suppose $e \in E_j$ for some $j < i$. Choose the largest such j. Then by property two, there is an $e' \in E_{j+1}$ such that $e' < e$, and by lemma 2 there is an $e'' \in E_{i+1}$ such that $e'' \leq e' < e$ contradicting property one.

Lemma 5: It is not possible that for infinitely many k's, $E_k \neq E_{k+1}$.

Proof: Suppose for infinitely many k's, $E_k \neq E_{k+1}$. Then (by lemmas 3 and 4) we can pick out an infinite sequence w_i of elements that are all distinct. By lemma 1 for i,j such that $i < j$, either $w_j < w_i$ or else w_i and w_j are incomparable. That is, the sequence w_i is an infinite nonincreasing sequence of distinct elements. But by a simple and well-known extension to the theorem that all sets of incomparable elements (in N^k) are finite, this is impossible.

Hence there exists k_1 such that $E_{k_1} = E_t$, for all $t \geq k_1$.

Lemma 6: There exists an integer $k \geq 0$ such that $E_k = E_{k+1}$ and $U_k = U_{k+1}$.

Proof: The sets U_i also satisfy properties 1 and 2, so the same proof implies that there exists k_2 such that $U_{k_2} = U_t$, for all $t \geq k_2$.

Therefore there exists k_3 such that $E_{k_3} = E_{k_3+1}$ and $U_{k_3} = U_{k_3+1}$.

Definition: Let k be the least integer such that $E_k = E_{k+1}$ and $U_k = U_{k+1}$.

Let $E = E_k$ and $U = U_k$.

Proposition 7: $L(E_i;U_i) \subseteq L(E_{i+1};U_{i+1})$; $E_i' \subseteq L(E_{i+1};U_{i+1})$.

Proof: By construction of the sets E_{i+1} and U_i', if $e \in (E_i \cup E_i')$, then for some $f \in E_{i+1}$ and $g \in U_i'$, $e \equiv fg$.

By definition of the sets U_i'' and U_{i+1} and the fact that if a set W of words is closed under quotients, $L(0; \min(W)) \supseteq W$, if $g \in U_i''$, then $g \in L(0;U_{i+1})$. Since $U_i \subseteq U_i' \subseteq U_i''$ this is also true if $g \in U_i$ or $g \in U_i'$.

Hence $e \in L(f;U_{i+1}) \subseteq L(E_{i+1};U_{i+1})$; therefore $E_i \cup E_i' \subseteq L(E_{i+1};U_{i+1})$. Hence $L(E_i \cup E_i';U_i) \subseteq L(E_{i+1};U_{i+1})$. In particular $L(E_i;U_i) \subseteq L(E_{i+1};U_{i+1})$, $E_i' \subseteq L(E_{i+1};U_{i+1})$.

Proposition 3: The equation C = D is satisfied in K(G,M) iff $D \in L(E;U)$.

Proof: 1. Since the quotient of two equivalent words is a unit (of the words) and the quotient of two units (of a word) is also a unit (of the word), all words in E are equal to the word C, while all words of U are units of the word C. Hence if D is in the semilinear set generated by E and U, then C = D.

2. Assume that C = D in K(G,M). Then there exists distinct words $C \equiv F_1, F_2, \ldots, F_k \equiv D$ such that

$$F_j \equiv F_j'A_{i_j} = F_j'B_{i_j} \equiv F_{j+1} \quad 1 \leq j \leq k-1,$$

where $A_{i_j} = B_{i_j}$ is an equation of M.

We will show by induction on j that $F_j \in L(E;U)$. For j = 1 this follows from the fact that $\{C\} = E_0$ and proposition 2. Assume now that $F_j \in L(E;U)$. Then $F_j \equiv e_1 u_1 \cdots u_t$ where $e_1 \in E$ and $u_\ell \in U$, $1 \leq \ell \leq t$. Consider now the least i in which all of e_1 and u_ℓ has appeared in the sets E_i, U_i in our construction. Consider the equation $F_j \equiv F_j'A_{i_j} = F_j'B_{i_j} \equiv F_{j+1}$. Note of course that we need to use at most α units in the formation of A_{i_j}; say the ones used were

$u_1, \ldots, u_s$ $(s \leq \alpha)$. But, by construction of E_i', a word Y that is the result of applying a rule to $e_1 u_1 \cdots u_s$ is a member of E_i'; note also that $u_{s+1}, \ldots, u_t$ are in U_{i+1} (since they are minimal). Hence by proposition 2 $F_{j+1} \in L(E_{i+1}; U_{i+1})$ and hence $F_{j+1} \in L(E;U)$.

Q.E.D.

From this constructive proof and the fact that the uniform membership problem for semilinear sets is decidable we obtain:

Corollary: The word problem for commutative semigroups is decidable.

Theorem 2: The fixed word equivalence problem for finitely presented commutative semigroups (ref. section 1) may be solved in real time on a Turing machine.

Proof: By a theorem of Fischer, Meyer, and Rosenberg [13] and Laing [29] the elements of a fixed semilinear set (using the notation referred to in section 1) can be recognized in real time on a Turing Machine (in fact by a Turing machine which manipulates its worktapes only as a counter machine does). The result follows by Theorem 1.

Recall that we have identified a word given over n generators in a commutative semigroup with a n-tuple of nonnegative integers. We shall identify the pair of words X,Y with the 2n-tuple in which X is identified with the first n-tuple and Y is identified with the second n-tuple.

The following theorem is due to M.A. Taiclin.

Theorem 3: The set of pairs of equivalent words of a finitely presented commutative semigroup is an effectively constructable

semilinear set.

Proof: Taiclin [45],[46],[48] has shown that the set of 2n-tuples of pairs of equivalent words is definable by a Presburger formula with 2n free variables and that this Presburger formula is effectively constructable. But Ginsburg and Spanier [16] have shown that the set definable by a Presburger formula is an effectively constructable semilinear set.

Theorem 4: The word problem for a fixed finitely presented commutative semigroup (ref. section 1) may be solved in real time on a Turing machine.

Proof: Same as Theorem 2.

Remark: If one uses an exponent notation for the fixed word equivalence problem or the word problem for fixed commutative semigroups (see the remarks at the end of section one), one may show the following results:

Theorem: The fixed word equivalence problem may be solved in linear time on a Turing machine. The word problem for a fixed finitely presented commutative semigroup may be solved in linear time on a Turing machine.

These results follow since one can recognize the elements of a fixed semilinear set (using this notation) in linear time on a Turing machine. (This follows by modifying Laing's proof.)

Section 4 - Related Algorithmic Problems

In this section we first study the computational complexity of some other algorithmic problems for commutative semigroups. We then briefly consider the complexity of some problems for vector addition systems.

Boundedness Problem

Consider the problem of determining whether or not a word has a finite or bounded number of words equivalent to it in a presentation of a commutative semigroup. There are again two problems which we can consider, namely: 1. the boundedness problem for a word in a fixed presentation of a commutative semigroup, that is, given an arbitrary word X and a fixed presentation P_0, are finitely or infinitely many words equivalent to X in P_0?; and 2. the uniform boundedness problem, i.e. given an arbitrary word X and an arbitrary presentation P, are finitely or infinitely many words equivalent to X in P?.

Recall from section 3 the identification we made between words given over n generators with n-tuples of nonnegative integers, and the identification of pairs of words with 2n-tuples of nonnegative integers.

Let S be a set of n-tuples. Let I_1, I_2 be a partition of $\{1,2, \ldots,n\}$ (i.e. $I_1 \cup I_2 = \{1,2, \ldots,n\}$, $I_1 \cap I_2 = \emptyset$). Say $I_1 = \{i_1,i_2, \ldots,i_k\}$ and $I_2 = \{j_1,j_2, \ldots,j_{n-k}\}$, where $i_1 < i_2 < \ldots < i_k$ and $j_1 < j_2 < \ldots < j_{n-k}$.

The (domain) projection of S onto I_1 is denoted $\pi_{I_1}(S)$

and is defined to be

$$\{(s_{i_1}, s_{i_2}, \ldots, s_{i_k}) \mid (s_1, s_2, \ldots, s_n) \in S\}.$$

For any $\vec{a} = (a_1, \ldots, a_{n-k})$ the projection of S onto I_1 restricted by $\vec{a}$ is denoted $\pi_{I_1}^{\vec{a}}(S)$ and is defined to be

$$\pi_{I_1}(\{(s_1, s_2, \ldots, s_n) \in S \mid (\forall \ell \leq n-k)[s_{j_\ell} = a_\ell]\}).$$

Theorem 1: Consider a presentation P_0 of a commutative semigroup. The set of words which have an infinite number of words equivalent to them in P_0 is an effectively constructable semilinear set.

Proof: Let the presentation P_0 have n generators. By Theorem 3 of section 3 we know that the set of pairs of equivalent words in P_0 is an effectively constructable semilinear set of 2n-tuples. Let S be that set, and let $S = L_1 \cup \ldots \cup L_k$, where L_i is a linear set.

Let (X,Y) denote the 2n-tuple identified with the pair of words X,Y. Let (0,Z) denote the 2n-tuple which contains zeroes in the first n-tuple and whose second n-tuple is the n-tuple identified with word Z.

A word X has an infinite number of words equivalent to it iff there exists at least one linear set L_i such that $(X,Y) \in L_i$ for some word Y and L_i has a period of the form (0,Z), for some word Z.

Let $L_{i_1}, \ldots, L_{i_j}$ be the linear sets which have a period of the form (0,Z).

Let $S' = L_{i_1} \cup L_{i_2} \cup \ldots \cup L_{i_j}$.

Apply a (domain) projection to S' onto the first n coordinates.

Call the resulting set S"; by results of Ginsburg and Spanier [15], S" is an effectively constructable semilinear set. S" is the desired set.

Corollary: The boundedness problem for a fixed presentation of a commutative semigroup may be solved in real time on a Turing machine.[1.]

Theorem 2: The uniform boundedness problem requires space at least proportional to n/logn on a multitape Turing machine.

Proof: Modify the construction given in section one for the lower bound by adding the additional state rule $Q_a \leftrightarrow Q_a Q_a$. Then the input $\mu(x)$ is accepted by the counter machine iff $I_x q_0 c_{n+1}$ has an infinite number of words equivalent to it.

Regularity Problem

Let S be a semigroup. We shall say that an element $a \in S$ is regular iff there exists $x \in S$ such that $a = axa$. We shall say that a word A is regular iff the element which A represents is regular. A semigroup S is regular iff all elements of S are regular.

The question of regularity is concerned with the existence of partial inverses in semigroups and is of interest in the theory of semigroups (see Lyapin [32] for examples and further details).

We will be concerned with the question of testing whether a word A is regular. As usual, we consider both the problem for fixed presentations and the problem for arbitrary presentations.

Theorem 3: Consider a presentation P_0 of a commutative semigroup. The set of words which are regular in the semigroup given by P_0 is

---

1. We assume the notation defined at the beginning of section one.

an effectively constructable semilinear set.

Proof: Let the presentation P_0 have n generators. Let S be the effectively constructable semilinear set of pairs of equivalent words in P_0. Let $S = L_1 \cup \ldots \cup L_k$, where L_i is a linear set.

Let $I^{(i)}$ be the initial vector of L_i and $\{P_1^i, \ldots, P_{n_i}^i\}$ the set of periods of L_i.

Let $I^{i,1}$ denote the first n-tuple of the 2n-tuple $I^{(i)}$ and $I^{i,2}$ denote the second n-tuple of $I^{(i)}$.

Similarly let $P_j^{i,1}$ denote the first n-tuple of P_j^i and $P_j^{i,2}$ denote the second n-tuple of P_j^i.

Fix i. Consider the set of simultaneous linear diophantine inequalities over the variables t_j:

$$I^{i,2} + \sum_{j=1}^{n_i} P_j^{i,2} t_j \geq 2(I^{i,1} + \sum_{j=1}^{n_i} P_j^{i,1} t_j) \qquad (*)$$

The set of nonnegative integer solutions to a set of simulataneous linear diophantine inequalities is an effectively constructable semilinear set (see van Leeuwen [52]).

Let S^i be the semilinear set for the set of diophantine inequalities (*). Let $S^i = L_1^i \cup \ldots \cup L_{m_i}^i$, where L_j^i is a linear set. For $X = (a_1, \ldots, a_{n_i})$, define X' to be the n-tuple $\sum_{j=1}^{n_i} a_j P_j^{i,1}$. Define $(L_j^i)'$ to be the linear set whose initial vector is the ' of the initial vector of L_j^i and whose periods are the ' of the periods of L_j^i. Define $(S^i)' = (L_1^i)' \cup \ldots \cup (L_{m_i}^i)'$.

Let $T^i = L(I^{i,1}; (S^i)')$. By slightly extending a result in Ginsburg and Spanier [15], one can show that if E is a semilinear set

and I a nonnegative vector, that L(I;E) is a semilinear set. Hence T^i is a semilinear set.

Then $T = T^1 \cup \ldots \cup T^k$ is the desired set.

Corollary: The regularity problem for a fixed presentation of a commutative semigroup may be solved in real time on a Turing machine.

Theorem 4: The uniform regularity problem requires space at least proportional to n/logn on a multitape Turing machine.

Proof: Modify the construction given in section one by adding the state relation $Q_a \leftrightarrow I_x q_0 c_{n+1} I_x q_0 c_{n+1} Q_a$. Then the word $I_x q_0 c_{n+1}$ is regular iff the counter machine accepts input $\mu(x)$.

Definition: Let s_1, s_2 be elements of a commutative semigroup; we shall say that s_2 divides s_1 iff $\exists s_3 \in S$ such that $s_1 = s_2 s_3$. We shall say that a word S_2 divides a word S_1 iff the element which S_2 represents divides the element which S_1 represents. Equivalently, we can say that a word S_2 divides a word S_1 iff there exists a word S' such that $S_1 = S'$ in the commutative semigroup and $S' > S_2$.[1] Divisibility is closely related to the coverability relation defined in Karp and Miller[25]; i.e. a word S_1 covers a word S_2 iff there exists a word S' such that $S_1 = S'$ in the commutative semigroup and $S' \geq S_2$.

Since regularity is a special case of either the divisibility or the coverability problem (namely take $S_2 = S_1 S_1$) we immediately have

---

1. Let X and Y be words. Let I_X and I_Y be the n-tuples identified with X and Y respectively. Then we shall say that $X > Y$ iff $I_X > I_Y$.

the following result:

Theorem 5: The uniform divisibility problem requires space at least proportional to n/logn on a multitape Turing machine; the uniform coverability problem requires space at least proportional to n/logn on a multitape Turing machine.

Theorem 6: Consider a presentation P_0 of a commutative semigroup. The set of pairs of words X, Y such that X covers Y in the presentation P_0 is an effectively constructable semilinear set.

Proof: Let S be the effectively constructable semilinear set of pairs of equivalent words in P_0.

Let $S = L_1 \cup \ldots \cup L_k$, where L_i is a linear set.

Let L_i have initial vector I^i and periods $\{P^i_1, \ldots, P^i_{n_i}\}$.

As before, let $I^{i,1}$ denote the first n-tuple of I^i, $I^{i,2}$ the second n-tuple of I^i, $P^{i,1}_j$ the first n-tuple of P^i_j, and $P^{i,2}_j$ the second n-tuple of P^i_j. Again let (X,Y) denote the 2n-tuple identified with the pair of words X,Y.

Let $\overline{I^i} = \{(I^{i,1}, J) \mid J \leq I^{i,2}\}$.

Let $\overline{P^i_j} = \{(P^{i,1}_j, Q) \mid Q \leq P^{i,2}_j\}$.

Let $\overline{L}_i = L(\overline{I^i}; \bigcup_{j=1}^{n_i} \overline{P^i_j})$.

Observe that $\overline{L_i}$ is an effectively constructable semilinear set.

Let $\overline{S} = \overline{L}_1 \cup \ldots \cup \overline{L}_k$

$\overline{S}$ is the desired set.

Corollary: The set of pairs of words X, Y such that Y divides X in presentation P_0 is an effectively constructable semilinear set.

Proof: Let S, $\overline{S}$ be as in the proof of the above theorem. Let S' be the effectively constructable semilinear set of those words which have only finitely many words equivalent to them in P_0. Let $S'' = S' \times N^n$.[1] Let $S''' = S'' \cap S$. Then $\overline{S} - S'''$ is an effectively constructable semilinear set [15] and is the desired set.

Corollary: The coverability problem for a fixed presentation may be solved in real time on a Turing machine; the divisibility problem for a fixed presentation may be solved in real time on a Turing machine.

We remark that the reachability tree construction (see Karp and Miller [25], Keller [26], or Hack [17]) may be used to uniformly solve the boundedness, regularity, divisibility, and coverability problems.[2] However the algorithm relies for termination upon incomparability in N^k and this reasoning is of the type that does not enable one to derive primitive recursive complexity bounds.

We note however that since the (uniform) boundedness, regularity, divisibility, and coverability problems are all efficiently reducible to the word problem and since the latter is solvable in exponential space, we have the following result:

Theorem 7: The uniform problems of boundedness, regularity, divisibility, and coverability may all be solved in exponential space on a Turing machine.

---

1. "X" denotes cartesian product.

2. Biryukov's algorithm may also be used to solve these problems.

Complexity of Word Problems for Abelian Groups

We now consider abelian groups and their presentations. Consider a set of generators $G = \{a_1, \ldots, a_n\}$ (1) and a set of defining relations of the form $A_i = B_i$, $1 \leq i \leq m$ (2), where A_i and B_i are words over G.

To the set of generators (1) we add the new generators $a_1^{-1}, a_2^{-1}, \ldots, a_n^{-1}$ (3) and to the defining relations (2), the new relations $a_i a_i^{-1} = \lambda$, $a_i^{-1} a_i = \lambda$, $1 \leq i \leq n$, (4), where λ denotes the empty word. As is well-known, the commutative semigroup given by generators (1) and (3) and defining relations (2) and (4) is an abelian group. This group we call the abelian group given by generators (1) and defining relations (2).

Note that words in an abelian group may be considered to have generators with negative coefficients. For a word X in an abelian group over generators (1), define X' to be the integer n-tuple $(x_1, \ldots, x_n)$, where $x_i = \#a_i$ in X.

Lemma 1: Let X and Y be words. X = Y in the abelian group given by (1) and (2) iff the set of linear diophantine equations $X' = Y' + \sum_{i=1}^{m} (B_i' - A_i')t_i$ has an integer solution $(t_1, t_2, \ldots, t_m)$.

Proof: Obvious.

Theorem 8: The uniform word problem for finitely presented abelian groups may be solved in polynomial time on a Turing machine.

Proof: We can test whether a set of linear diophantine equations has any integer solutions or not in polynomial time (See Knuth [28], Karp [24]). The theorem follows by the preceding lemma.

Corollary: The uniform word problem for commutative semigroups with cancellation may be solved in polynomial time on a Turing machine.

Proof: Let X and Y be words whose generators have only nonnegative exponents. According to a theorem in Lyapin [32], X = Y in the commutative cancellation semigroup given by (1) and (2) iff X = Y in the abelian group given by (1) and (2).

Theorem 9 (Tseiten): Let a commutative semigroup be given by generators (1) and defining relations (2). Let h be the maximium coefficient of a generator in the set of words $\{A_i, B_i\}$. Assume that the two words X and Y are such that the coefficients of all the generators of both words are greater than or equal to h(m+1). Then X = Y in the commutative semigroup given by (1) and (2) iff X = Y in the abelian group given by (1) and (2).

Proof: See Emelichev [10].

Theorem 9 says that there exists a polynomial time algorithm for a large subclass of the word problems for commutative semigroups.

I should remark that theorem 9 is of interest not only for this reason. Indeed it was a basic lemma in Taiclin's proof of Theorem 3, section 3. It is also of importance in obtaining polynomial time algorithms for many other algorithmic problems for commutative semigroups.

Polynomial Time Algorithms for Commutative Semigroups

Unit Semigroup Problem

Consider the problem of determining whether an arbitrary presentation defines the unit semigroup[1] (equivalently whether a word is equivalent to all other words in that presentation).

Theorem 10: The problem of determining whether an arbitrary presentation defines the unit semigroup may be solved in polynomial time on a Turing machine.

Proof: Let the presentation have generators $G = \{a_1, \ldots, a_n\}$ and defining relations $A_j = B_j$, $j = 1, \ldots, m$. We seek to determine whether $a_1 = a_2 = \ldots = a_n$ and $a_1 a_1 = a_1$.

Define the canonical set C_i of a generator a_i as the minimal set $\subseteq G$ such that if X is a word such that $a_i^q = X$ for some power q then the generators contained in X are contained in C_i. By using a simple algorithm first observed by Biryukov [5] we can determine what C_i is in polynomial time.

Clearly C_i must equal G for each i if the semigroup is to be unit.

It is also clear that if the semigroup is to be unit, that each a_i must be regular. There is an algorithm in Emelichev [11], which given a generator a_i will:

1. if the generator a_i is nondegenerate[2] it will determine

---

1. The unit semigroup is the semigroup consisting of one element.

2. A generator a_i is nondegenerate if no equation of the form $a_i = N$ holds, where N is a word that does not contain a_i. A generator is degenerate if it is not nondegenerate.

whether a_i is regular or not.

2. if the generator a_i is degenerate, then it will either determine whether a_i is regular or not, or it will determine that a_i is degenerate and will find a word N such that $a_i = N$, where $\neq a_i$ in $N = 0$. In case a_i is degenerate the algorithm will then eliminate a_i from the presentation by substituting N for a_i in all relations that contain a_i.

By making modifications to this algorithm, this algorithm will run in polynomial time.

Now, since $C_i = G$, there exists q such that $a_i^q = X$, where the word X contains every generator. Furthermore since a_i is regular, an equation $a_i = a_i^2 Y$ holds, where Y is some word. Let h equal the maximum coefficient of a generator in the words $\{A_j = B_j \mid j = 1, \ldots, m\}$. Let $r = h(m+1)$.

$$\text{Then } a_i = a_i^{qr} Y^{qr-1}$$

$$= X^r Y^{qr-1}.$$

Hence a_i is equal to a word where the coefficient of each generator in the word is greater than or equal to r.

Let X_0 be the word $a_1^r a_2^r \cdots a_n^r$.

Hence to test whether $a_1 = a_2 = \ldots = a_n$ and $a_1 a_1 = a_1$ it suffices to test whether all words greater than X_0 are equal. But by Tseiten's lemma this is true iff all words greater than X_0 are equal in the corresponding abelian group which is true iff the corresponding group consists of one element. But we may test whether an abelian group is a unit group in polynomial time; i.e. by testing whether $a_1 = a_2 = \ldots = a_n$

and $a_1 a_1 = a_1$.

We point out now that several algorithmic problems for commutative semigroups can be solved in polynomial time on a Turing machine. In particular each of the following problems can be solved in polynomial time:

1. Whether an arbitrary word in an arbitrary presentation has finite or infinite type (order). [An element s of a semigroup has infinite type iff each of the elements s^i ($i \geq 1$) are different; an element s has finite type iff it does not have infinite type (i.e. iff there exists i, j, $i \neq j$, such that $s^i = s^j$). A word has the type of the element which it represents.][1].

2. Whether the semigroup denoted by an arbitrary presentation has a finite number of elements or not.

3. Whether a nondegenerate generator in an arbitrary presentation is regular or not.[2].

Whether the semigroup denoted by an arbitrary presentation is regular or not.

4. Whether the semigroup denoted by an arbitrary presentation has a zero or not. [$z \in S$ is a zero of a commutative semigroup iff $zs = z$, for all $s \in S$.]

5. Whether the semigroup denoted by an arbitrary presentation

---

1. We remark that the question of whether there exists $j > 1$ such that $s^1 = s^j$ can be shown to require space at least proportional to n/logn on a Turing machine.

2. We remark that the question of whether a degenerate generator is regular or not can be shown to require space at least proportional to n/logn.

is nilpotent or not. [A semigroup S with zero z is said to be nilpotent iff there exists a positive integer k such that $s^k = z$ for all $s \in S$.]

6. Whether the semigroup denoted by an arbitrary presentation is a group or not.

7. Whether the semigroup denoted by an arbitrary presentation has an identity element or not. [i is an identity element of S iff is = s for all $s \in S$.]

Whether a nondegenerate generator in an arbitrary presentation has a unit or not.[1.]

These results follow for the main part by an analysis of the algorithms given in Biryukov[5] and Emelichev[9], [10], [11]. However in certain cases (in particular for problems 1,2,3,6, and 7) certain modifications have to be made to the algorithms in order that they run in polynomial time.

Vector Addition Systems

We now briefly consider the complexity of some problems for vector addition systems (V.A. systems).

We shall assume that n-tuples are represented using binary

---

1. We remark that the question of whether a degenerate generator has a unit or not can be shown to require space at least proportional to n/logn.

notation for the component integers. That is, we represent a vector or n-tuple as $(a_1, a_2, \ldots, a_n)$, where a_i is an integer represented in binary.

We define a symmetric vector addition system as a vector addition system $V = < I, P >$ in which $\forall p \in P, -p \in P$. By the remarks concerning alternate notation at the end of section one and by use of the correspondence between vector addition systems and commutative semiThue systems, we immediately have the following result:

Theorem 11: The uniform reachability problem for symmetric vector addition systems requires space at least proportional to n on a multitape Turing machine.

Define the uniform membership problem for linear sets as follows: Given an arbitrary vector J, an arbitrary vector I, and an arbitrary set of periods P, is $J \in L(I;P)$? We assume as in vector addition systems binary notation for the component integers.

Let NP denote the class of languages which can be accepted in nondeterministic polynomial time on a Turing machine.

Lemma: The uniform membership problem for linear sets is NP-complete.[1.]

Proof: 1. We first prove that the problem is NP-hard. We do this by showing that a known NP-hard problem can be efficiently reduced to it.

---

1. We refer the reader to Karp [24] for definitions of the terms NP-complete and polynomial time reducibility. This lemma was observed in conversations with A.R. Meyer.

Definition: The Subset Sum Problem is defined as follows:

Given an input $(k_1, k_2, \ldots, k_{n+1})$ of n+1 nonnegative integers (represented in binary), does there exist a set $S \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{j \in S} k_j = k_{n+1}$.

The Subset Sum Problem is known to be NP-hard. We now show that we can efficiently reduce the Subset Sum Problem to the uniform membership problem for linear sets.

Given an input $(k_1, k_2, \ldots, k_{n+1})$, construct vector $J = (1, 1, \ldots, 1, k_{n+1})$, initial vector $I = (0, 0, \ldots, 0)$, and the set P of 2n periods:

$\{(1, 0, \ldots, 0, k_1), (1, 0, \ldots, 0, 0), (0, 1, 0, \ldots, 0, k_2),$

$(0, 1, 0, \ldots, 0, 0), \ldots, (0, \ldots, 0, 1, k_n), (0, \ldots, 0, 1, 0)\}$ (that is, P contains those periods which have a 1 in the ith position, k_i in the n+1st position and zeroes everywhere else, and those which have a 1 in the ith position and zeroes everywhere else).

Then there exists a subset $S \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{j \in S} k_j = k_{n+1}$ iff $J \in L(I;P)$.

Since this reduction can be constructed in polynomial time, the uniform membership problem for linear sets is NP-hard.

2. We now show that the uniform membership problem for linear sets is in NP.

We can solve the uniform membership problem of linear sets by determining whether a set of linear diophantine equations has any nonnegative integer solutions or not; that is, to test whether $J \in L(I;P)$ where $P = \{P_1, \ldots, P_m\}$ we simply have to determine whether

$$J = I + t_1P_1 + \ldots + t_mP_m$$

$$\text{i.e. } J - I = t_1P_1 + \ldots + t_mP_m$$

has any nonnegative integer solutions. By a recent result of Sieveking [43], we can determine whether a set of linear diophantine inequalities has any integer solutions or not in nondeterministic polynomial time, and hence can determine whether a set of linear diophantine equations has any nonnegative integer solutions or not in nondeterministic polynomial time. □

Since linear sets are simply the subcase of vector addition systems in which the periods are all nonnegative, we can immediately conclude that a very special subcase of the reachability problem is at least NP-hard.

Remark: We shall occasionally consider vector addition system without a fixed initial vector. That is, we shall sometimes consider vector addition systems as a set of periods, and be concerned with such questions as: is vector J reachable from vector I in the vector addition system $V = \langle P \rangle$, where P is a set of periods.

Definition

Let V be a vector addition system. Let * represent the don't care symbol. We shall say that a function $f: N \rightarrow N$ is weakly computable by V if given an initial vector I with x in a specified input coordinate and a standard initial marking in the other coordinates if for every $0 \leq y \leq f(x)$ there is a vector reachable from I of the form $(*, *, \ldots, *, y)$, and conversely that every vector reachable from I has the form $(*, *, \ldots, *, y)$, where $0 \leq y \leq f(x)$. We define "weakly computable" by a commutative semigroup similarly; that is, a function

f: $N \to N$ is weakly computable by a commutative semigroup if given a word $X = a_1^x Z$, where Z is some fixed word over generators $\{a_2, \ldots, a_n\}$, if for every $0 \leq y \leq f(x)$ there is a word Y equivalent to X where $\#a_n$ in Y = y, and conversely that every word Y equivalent to X has $\#a_n$ in Y = y, where $0 \leq y \leq f(x)$.

Michel Hack [18] has shown

Theorem 12: For any n, there is a vector addition system which weakly computes a function at least as large as the nth Ackermann function.

One can in fact show that vector addition systems can exactly weakly compute the 2nd and 3rd Ackermann functions: squaring and exponentiation. It appears that V.A. systems can exactly compute the nth Ackermann function, although the details for this have not been verified yet.

Theorem 12 has been used to show that there are vector addition systems whose reachability sets, although finite, grow as large as Ackermann's function of the size of the vector addition system.

However:

Theorem 13: $f(x) = x^2$ is not weakly computable by a commutative semigroup.

Proof: Assume that f is weakly computable by a commutative semigroup presentation P with n generators. Recall that the set of pairs of equivalent words of P is a semilinear set S of 2n dimensions. Then the projection of S onto coordinates $\{1, n+1, n+2, \ldots, 2n\}$ restricted by the standard marking on coordinates $\{2, \ldots, n\}$ is a semilinear set S'. Now apply a (domain) projection to S' on the first and last

coordinates. The result is $\{(x,y) \mid y \leq x^2\}$ is semilinear. But this is impossible, since according to Ginsburg and Spanier [16], $\{(x,y) \mid y \leq x^2\}$ is not a semilinear set.

There thus is a substantial difference between the functions which vector addition systems can weakly compute and the functions which commutative semigroups can weakly compute.

Boundedness and Coverability for Vector Addition Systems

Let V be a vector addition system and I a nonnegative vector. Consider the following problems:

1. Boundedness Problem: Is the reachability set R(I) finite?

2. Coverability Problem: Given a vector J, does there exist $K \in R(I)$ such that $K \geq J$?

By a fixed vector addition system V_0, we shall mean a fixed set of periods P_0. By the boundedness problem for a fixed vector addition system we shall mean the following problem: Given a vector I, is the reachability set R(I) finite in the vector addition system V_0?

Lemma: Consider a fixed vector addition system V_0. The set of vectors with infinite reachability sets in V_0 is a semilinear set.

Proof(Nonconstructive): Note that if a vector I has an infinite reachability set, then all vectors greater than I also have infinite reachability sets.

Let T equal the min of the set of vectors that have infinite reachability sets.

T is finite.

Let $S = \{J \mid J$ is a vector and $\exists K \in T$ such that $J \geq K\}$.

S is semilinear and is the desired set.

Corollary: The set of vectors with finite reachability sets in a fixed vector addition system V_0 is a semilinear set.

Proof: The complement of a semilinear set is a semilinear set [15].

Corollary: The boundedness problem for a fixed vector addition system may be solved in linear time on a Turing machine.

Lemma: Consider a fixed vector addition system V_0. The set of pairs of vectors X, Y such that X covers Y in V_0 is not in general a semilinear set.

Corollary: Consider a fixed vector addition system with a fixed initial vector I_0. The set of vectors X such that I_0 covers X is not in general a semilinear set.

Proofs: This follows by a technique similar to the one Rabin used in [1] to show that reachability sets of vector addition systems are not in general semilinear. See [1] for further details.

We remark:

Lemma: Consider a fixed vector addition system V_0 with a fixed initial vector I_0. The set of vectors X such that X covers I_0 is a semilinear set.

Lemma: Consider a fixed vector addition system V_0 with a fixed initial vector I_0. The set $\{J \mid J \in R(I_0)$ and J has a finite reachability set$\}$ is not in general a semilinear set. The set $\{J \mid J \in R(I_0)$ and J has an infinite reachability set$\}$ is not in general a semilinear set.

It is known that by use of the reachability tree construction [25] that one may uniformly solve either the boundedness problem or the coverability problem for vector addition systems. We remark that by the correspondence between vector addition systems and commutative semigroups:

Theorem: The uniform boundedness problem for symmetric vector addition systems requires space at least proportional to n on a Turing machine.

The uniform coverability problem for symmetric vector addition systems requires space at least proportional to n on a Turing machine.

Remark added: R.J. Lipton [31] has just recently shown that the uniform reachability and boundedness problems for vector addition systems requires exponential space on a Turing machine. However Lipton's proof does not (immediately) extend to the case of symmetric vector addition systems.

Definition:

Let X and Y be two arbitrary words and let P and Q be two arbitrary commutative semiThue systems. The uniform equivalence problem is defined as follows: Is the set of words derivable from X in commutative semiThue system P the same as the set of words derivable from Y in commutative semiThue system Q? The uniform inclusion problem is: Is the set of words derivable from X in P contained in the set of words derivable from Y in Q? These problems have been shown to be unsolvable by Hack [20] and Rabin [1].

Remark: Although at first glance it might appear that the word problem problem for commutative semigroups and the derivation problem for

commutative semiThue systems are similar, in fact it appears likely that there is a significant difference between them. For example, it is known that the uniform equivalence and inclusion problems are unsolvable for commutative semiThue systems, while these problems are solvable for commutative semigroups. We have also seen that commutative semiThue systems can exactly weakly compute functions at least as large as the nth Ackermann function, while commutative semigroups cannot even weakly compute $f(x) = x^2$. However these differences have been for sets of words and no one at present (to the author's knowledge) has been able to prove that such a difference exists for the word problem itself.

Remark: Recall Tseiten's example (in section one) of a simple presentation with an unsolvable word problem. One might ask if one can construct a simple presentation that is almost commutative (in the sense that it lacks the relation ab = ba for only one pair of generators a, b) but that has an unsolvable word problem. In fact it turns out that it is possible to construct such presentations, but for uninteresting reasons. For, by a simple observation of Hall's [21], it turns out that given a presentation with m generators and r relations with an unsolvable word problem, that one can construct a presentation with 2 generators and the same number (r) of relations that is also unsolvable.

BIBLIOGRAPHY

1. Baker, H., "Rabin's Proof of the Undecidability of the Reachability Set Inclusion Problem of Vector Addition Systems," Computation Structures Groups Memo 79, Project MAC, M.I.T. 1973.

2. Baker, H., "Vector Addition System Reachability Sets Are Almost Semilinear," Unpublished Manuscript.

3. Biryukov, A.P., "Solvability of the Problem of Isomorphism for Finitely Defined Commutative Semigroups with Two Generators," Interuniv. Science Symposium on General Algebra, pp. 5-6.

4. Biryukov, A.P., "Solution of Certain Algorithmic Problems for Finitely Determined Commutative Semigroups," Seventh All-Union Colloquium on General Algebra, Kishinev, 1965.

5. Biryukov, A.P., "Solution of Certain Algorithmic Problems for Finitely Determined Commutative Semigroups," Siberian Mathematics Journal, 1966, Vol.7, No.3, pp 423-428 .

6. Biryukov, A.P., "Some Algorithmic Problems for Finitely Defined Commutative Semigroups," Siberian Mathematics Journal, Vol.8 , 1967, pp. 384-391.

7. Carlisle, W H., "Residual Finiteness of Finitely Generated Commutative Semigroups," Pacific Journal of Mathematics. Vol. 36, 1971, pp. 99-101.

8. Emelichev, V.A., "Commutative Semigroups with One Defining Relation," Shuya Gosudarstvennyi Pedagogicheskii Institut Uchenye Zapiski, Vol. 6, 1958, pp. 227-242.

9. Emelichev, V.A., "Solution of Some Algorithmic Problems for Commutative Semigroups," Soviet Mathematics Doklady, Vol.3, 1962, pp. 699-702.

10. Emelichev, V.A., "Algorithmic Solvability of Certain Mass Problems in the Theory of Commutative Semigroups," Sibirski Matem. Zh.,1963, Vol. 4, No.4, pp. 788-798.

11. Emelichev, V.A.,"An Algorithm for Discerning Regularity of a Finitely Defined Commutative Semigroup," Dokl. Akad. Nauk BSSR, 1965, Vol.9, pp. 713-716.

12. Evans,T., "Some Connections between Residual Finiteness, Finite Embeddability, and the Word Problem," Journal of the London Math. Soc., 1969, Vol.2, pp. 399-403.

13. Fischer, P.C., Meyer, A.R., and A.L. Rosenberg, "Counter Machines and Counter Languages," Mathematical Systems Theory, Vol.2, No. 3, pp. 265-283.

14. Freyd, P., "Redei's Finiteness Theorem for Commutative Semigroups," Proc. American Math. Soc. 1968, Vol. 19, pg. 1003.

15. Ginsburg, S. and E.H. Spanier, "Bounded Algol-like Languages," Trans. Amer. Math. Soc., 1964, Vol. 113, pp. 333-368.

16. Ginsburg, S. and E.H. Spanier, "Semigroups, Presburger Formulas, and Languages," Pacific Journal of Mathematics, 1965, Vol.16, No.2, pp. 285-296.

17. Hack, M., "Decision Problems for Petri Nets and Vector Addition Systems," Computation Structures Group Memo 95, Project MAC, M.I.T., 1974.

18. Hack, M., "The Recursive Equivalence of the Reachability Problem and the Liveness Problem for Petri Nets and Vector Addition Systems," 15th Annual Symposium on Switching and Automata Theory, 1974, pp. 156-164.

19. Hack, M., "Petri Nets and Commutative Semigroups," Computation Structures Note 18, Project MAC, M.I.T., 1974.

20. Hack, M., "The Equality Problem for Vector Addition Systems is Undecidable," Computation Structures Group Memo 121, Project MAC, M.I.T., 1975.

21. Hall, M., "The Word Problem for Semigroups with Two Generators," J. Symbolic Logic, Vol. 14, 1949, pp. 115-118.

22. Hermann, G., "Die Frage der Endlich Vielen Schritte in der Theorie der Polynomideale," Math. Ann., Vol. 95, 1926, pp.736-788.

23. Hopcroft, J.E. and J.D. Ullman, Formal Languages and Their Relation to Automata, Addison-Wesley Publishing Company, 1969.

24. Karp, R., "Reducibility among Combinatorial Problems," in Complexity of Computer Computations, ed. Miller and Thatcher, Plenum Press, New York, 1972.

25. Karp, R. and R. Miller, "Parallel Program Schemata: A Mathematical Model for Parallel Computation," Switching and Automata Theory Symposium, 1967, pp. 55-61.

26. Keller, R.M., "Vector Replacements Systems: A Formalism for Modeling Asynchronous Systems," Technical Report 117, Department of Elec. Eng., Princeton University, 1972.

27. Khalelov, E.A., "Algorithmic Solutions of Certain Problems in the Theory of Commutative Semigroups with Cancellation," Siberian Mathematics Journal, Vol.7, 1966, pp. 343-356.

28. Knuth, D., The Art of Computer Programming, Vol.2, Seminumerical Algorithms, Addison-Wesley Publishing Company, 1969.

29. Laing, R., "Realization and Complexity of Commutative Events," University of Michigan Technical Report 035105-48-T.

30. Lallemont, G., "On a Theorem of Malcev," Proc. Amer. Math. Soc., Vol. 30, No. 1, 1971, pp. 49-54.

31. Lipton, R.J., "The Reachability Problem Requires Exponential Space," Extended Abstract, 1975.

32. Lyapin, E.S., Semigroups, Moscow, 1958.

33. Malcev, A.I., "On Homomorphisms of Finite Groups," Ivano Gosudarstvennyi Pedagogicheskii Institut Uchenye Zapiski, Vol. 18, 1958, pp. 49-60.

34. Markov, A.A., "The Impossibility of Certain Algorithms in the Theory of Associative Systems," Dokl. Akad. Nauk SSSR, Vol. 55, 1947, pp. 587-590.

35. Matijasevic, Ju. V., "Simple Examples of Undecidable Associative Calculi," Soviet Math. Doklady, Vol. 8, 1967, pp. 555-557.

36. Meyer, A.R., "Weak Monadic Second Order Theory of Successor is not Elementary-Recursive," Project MAC Tech. Memo. 38, 1973.

37. Post, E., "Recursive Unsolvability of a Problem of Thue," J. Symbolic Logic, Vol. 12, 1947, pp. 1-11.

38. Rabin, M., Logic, Automata, and Computability Conference, Oriskany, New York, 1965.

39. Rackoff, C. Personal Communication.

40. Redei, L., The Theory of Finitely Generated Commutative Semigroups, Pergamon Press, New York, 1965.

41. Scott, D., "A Short Recursively Unsolvable Problem," J. Symbolic Logic, Vol. 21, 1956, pp. 111-112.

42. Seidenberg, A., "Constructions in Algebra," Trans. American Mathematical Society, Vol. 197, 1974, pp. 272-313.

43. Sieveking, Unpublished Manuscript.

44. Stockmeyer, L., "The Complexity of Decision Problems in Automata Theory and Logic," Project MAC Tech. Report 133, 1974.

45. Taiclin, M.A., "On Elementary Theories of Commutative Semigroups with Cancellation," Algebra i Logika, Vol. 5, No. 1, 1966, pp. 51-69.

46. Taiclin, M.A., "On Elementary Theories of Commutative Semigroups," Algebra i Logika, Vol. 5, No. 4, 1966, pp. 55-89.

47. Taiclin, M.A., "Two Remarks on Isomorphisms of Commutative Semigroups," Algebra i Logika, Vol. 6, 1967, pp. 95-110.

48. Taiclin, M.A., "Algorithmic Problems for Commutative Semigroups," Soviet Mathematics Doklady, Vol. 9, 1968, pp. 201-204.

49. Taiclin, M.A., "On the Isomorphism Problem for Commutative Semigroups," Siberian Mathematics Journal, Vol. 9, 1968, pp. 375-401.

50. Taiclin, M.A., "The Isomorphism Problem for Commutative Semigroups," Mat. Sb. (N.S.), 93 (135), 1974, pp. 103-128, 152.

51. Tseiten, G.S., "Associative Calculus with Insolvable Equivalence Problem," Dokl. Akad. Nauk SSSR, Vol. 107, 1956, pp. 370-371.

52. van Leeuwen, J., "A Partial Solution to the Reachability Problem for Vector Addition Systems," 6th Annual A.C.M. Symposium on the Theory of Computing, 1974, pp. 303-309.