Acta Informatica 22, 421–432 (1985)

Complexity of the Word Problem for Commutative Semigroups of Fixed Dimension

Dung T. Huynh

Computer Science Department, Iowa State University, Ames, IA 50011, USA

Summary. In this paper we investigate the computational complexity of the word problem for commutative semigroups of fixed dimension. It is shown that for commutative semigroups of dimension k, $k \geqq 6$, the word problem is complete for symmetric linear space, providing another complete problem for this symmetric complexity class. We also show that in the case of one generator, the word problem is solvable in polynomial time.

0. Introduction

Decidability results for commutative semigroups have been derived by Russian mathematicians, including Biryukov, Taiclin (cf. [2, 12]). It is known that the (uniform) word problem and the equivalence problem are decidable. The word problem is the problem of determining whether two words of a given commutative semigroups are equivalent. In the equivalence problem it is to decide whether two equivalence classes of two given commutative semigroups are equal.

For the complexity theorist it is interesting to investigate the computational complexity of the above problems. The first significant result in this direction is the work of Cardoza, Lipton, Mayr and Meyer (cf. [1, 10]). They showed that the word problem is complete for exponential space, while the complexity of the equivalence problem is still an open question[1]. (See [10] for a detailed presentation of this result.)

In this paper we want to investigate the complexity of the word problem under the condition that the dimension of the commutative semigroup is bounded, i.e., we are interested in commutative semigroups with a bounded number of generators. What we expect is that bounding the number of generators may lower the complexity of the above decision problems. In fact, we will prove that the word problem for commutative semigroups with a bounded

[1] Recently, a complexity result for the equivalence problem has been obtained in [7]

number of generators is complete for symmetric linear space, showing an exponential difference to the uniform case.

This paper consists of 6 sections. Section 1 contains basic definitions, notations used later and statements of the results. In Sect. 2 we define the notion of symmetric Turing machines introduced in [9] by Lewis and Papadimitriou as a model for symmetric computation. Section 3 introduces symmetric counter machines without input terminals. The lower bound for the word problem is proved in Sect. 4. Section 5 shows the upper bound. Section 6 contains the result for commutative semigroups with one generator.

1. Preliminaries and Results

In this paper $\mathbb{N}$ denotes the set of nonnegative integers. Σ^* denotes the free monoid generated by the finite alphabet Σ. ε denotes the empty word. $|w|$ denotes the length of w, $w \in \Sigma^*$.

For a finite alphabet S, $S^\oplus$ denotes the free commutative monoid generated by S. If S has cardinality k, $S=\{s_1, \ldots, s_k\}$, then $S^\oplus$ is isomorphic to $\mathbb{N}^k$. An element x of $S^\oplus$ is written in the form $x=s_1^{e_1} \ldots s_k^{e_k}$, $e_1, \ldots, e_k \in \mathbb{N}$. The operation in $S^\oplus$ is denoted by $+$, e.g. $u+v$, where $u, v \in S^\oplus$.

A (semigroup) presentation of a finitely generated commutative semigroup is a pair (S, R), where S is the finite set of generating symbols and $R \subseteq S^\oplus \times S^\oplus$ is the finite set of defining relations. (S, R) represents the semigroup $S^\oplus/\sim$, where $\sim$ is the congruence defined by R.

It is well known that finitely generated commutative semigroups are finitely defined, i.e. they are presented by presentations with finitely many defining relations (cf. e.g. [3]). In this paper we only consider finite presentations.

Let (S, R) be a finite presentation. For $z, z' \in S^\oplus$ define

$$z \underset{R}{\leftrightarrow} z' \text{ iff there is a relation } (u, v) \in R \text{ such that either (1) } z=u+w \text{ and } z'=v+w \text{ or (2) } z'=u+\bar{w} \text{ and } z=v+\bar{w},$$

where $w, \bar{w} \in S^\oplus$. $\underset{R}{\leftrightarrow}$ is written as $\leftrightarrow$ if R is understood. Further let $\overset{*}{\underset{R}{\leftrightarrow}}$, or $\overset{*}{\leftrightarrow}$ for short, denote the reflexive and transitive closure of $\leftrightarrow$.

If $x \sim y \bmod R$, or $x \sim y$ for short, then there are $z_0, z_1, \ldots, z_n \in S^\oplus$ such that

$$x=z_0 \leftrightarrow z_1 \leftrightarrow z_2 \leftrightarrow \ldots \leftrightarrow z_n=y.$$

Such a sequence is called a derivation in (S, R).

The uniform word problem for finitely generated commutative semigroups of dimension $\leqq k$, $k \in \mathbb{N}$, is defined as follows.

Input: A presentation (S, R) and two words $x, y \in S^\oplus$, where $\text{card}(S) \leqq k$.

Question: Is $x \sim y \bmod R$?

We denote this problem by UWP(k).

Main Result. UWP(k) is complete for symmetric linear space for $k \geqq 6$.

For the case $k=1$ we also show that UWP(1) is solvable in polynomial time.

Of course, we have to define the size of an input instance. This can be done in a straightforward way: Encode the exponents of a word in binary notation. (Note that k is fixed.)

In subsequent sections we assume that the reader is familiar with standard notions in complexity theory (cf. [5]).

Before proceeding to prove our results, we give here the proof ideas. The upper bound of UWP(k) can be derived easily by using known results (cf. Sect. 5). The lower bound proof is based on several reductions: the reduction of linear-space-bounded symmetric Turing machines to exponential-space-bounded symmetric counter machines without input terminals, the reduction of exponential-space-bounded symmetric counter machines without input terminals to UWP(k).

2. Symmetric Turning Machines

In this section we introduce the notion of symmetric Turing machines and reproduce the fundamental lemma concerning symmetric computation. For more details the reader is referred to [9].

Symmetric Turing machines were introduced in [9] as a model for symmetric computation. In order to obtain symmetric Turing machines, an ordinary (nondeterministic) Turing machine M will be a 8-tuple $(Q, \Gamma, \Sigma, k, \Delta, q_0, F, \#)$, where

Q is the finite set of states,
Γ is the finite alphabet of tape symbols,
$\Sigma \subseteq \Gamma$ is the input alphabet,
$k>0$ is the number of tapes,
$q_0 \in Q$ is the initial state,
$F \subseteq Q$ is the set of final states,
Δ is a finite set of transitions, and
$\# \in \Gamma \setminus \Sigma$ is the blank symbol.

A transition of M is of the form $(p, t_1, \ldots, t_k, q)$, where $p, q \in Q$ and for all $i=1, \ldots, k$, t_i is either a triple of the form[2] (ab, D, cd) with $a, b, c, d \in \Gamma$ and $D \in \{+1, -1\}$, or a triple of the form (a, o, b) with $a, b \in \Gamma$.

The tapes of M are one-way infinite. A configuration of M is defined as usual. $\mathscr{C}(M)$ denotes the set of configurations of M. $\vdash_M$ (or $\vdash$ if M is understood) denotes the 'yield' relation on $\mathscr{C}(M)$. $\vdash_M^*$ denotes the reflexive and transitive closure of $\vdash_M$.

$\mathscr{I}(M)$ and $\mathscr{F}(M)$ denote the sets of initial and final configurations of M, respectively.

In order to obtain symmetric Turing machines, we need to define the inverse of a transition. Let $\delta=(p, t_1, \ldots, t_k, q)$ be a transition of M. Then the

[2] $(ab, +1, cd)$ means M, scanning the symbol a on the ith tape with b in the cell just right to the scanned cell, will change ab to cd and move the head to the right

inverse δ^{-1} of δ is $(q, t_1^{-1}, \ldots, t_k^{-1}, p)$, where $t_i^{-1} := (\beta_i, -D_i, \alpha_i)$ if $t_i = (\alpha_i, D_i, \beta_i)$, $1 \leqq i \leqq k$.

The inverse of a Turing machine $M = (Q, \Gamma, \Sigma, k, \Delta, q_0, F, \#)$ is $M^{-1} = (Q, \Gamma, \Sigma, k, \Delta^{-1}, q_0, F, \#)$, where $\Delta^{-1} := \{\delta^{-1} \mid \delta \in \Delta\}$. A Turing machine is symmetric if it is its own inverse. The symmetric closure of a Turing machine $M = (Q, \Gamma, \Sigma, k, \Delta, q_0, F, \#)$ is $\bar{M} = (Q, \Gamma, \Sigma, k, \Delta \cup \Delta^{-1}, q_0, F, \#)$.

Let $S: \mathbb{N} \to \mathbb{N}$ be a function. SSPACE($S(n)$) denotes the class of languages accepted by $S(n)$-space-bounded symmetric Turing machines. We will be interested in SSPACE(n). A language L is SSPACE(n)-complete if $L \in$ SSPACE(n) and every $L' \in$ SSPACE(n) is log-space reducible to L.

In showing our main result, we will employ Lewis and Papadimitriou's fundamental lemma (Lemma 1 in [9]) for symmetric computation. This lemma states 3 conditions under which a Turing machine can be symmetrically closed without affecting the accepted language and the space bound. For the sake of completeness, we reproduce this lemma here.

For every Turing machine M we define a Turing machine $M\#$ that satisfies the following properties:

(1) M and $M\#$ accept the same language within the same space bound, and have the same number of tapes;

(2) $M\#$ has no transitions into its initial state or out of any final state;

(3) $M\#$ never decreases the amount of cells used on any tape. (This can be accomplished by introducing a 'pseudo-blank' symbol.)

Let M be any Turing machine and $\mathscr{A} \subseteq \mathscr{C}(M)$ be a set of special configurations. Let $C_0, C_1, \ldots, C_n \in \mathscr{C}(M)$. If $n \geqq 0$, $C_0 \vdash C_1 \vdash \ldots \vdash C_n$, and if $n \geqq 1$ then $C_1, \ldots, C_n \notin \mathscr{A}$, then we write $C_0 \overset{*\mathscr{A}}{\vdash} C_n$. If $A_1, A_2 \in \mathscr{A}$ and there is $C \in \mathscr{C}(M)$ so that $A_1 \overset{*\mathscr{A}}{\vdash} C \vdash A_2$, then we write $A_1 \overset{+\mathscr{A}}{\vdash} A_2$ (or $A_2 \overset{+\mathscr{A}}{\dashv} A_1$). $A_1 \overset{+\mathscr{A}}{\vdash\!\!\dashv} A_2$ stands for $A_1 \overset{+\mathscr{A}}{\underset{\bar{M}}{\vdash}} A_2$.

Lemma 2.1 [9]. *Let $M = (Q, \Gamma, \Sigma, k, \Delta, q_0, F, \#)$ be any Turing machine, and let $\mathscr{A} \subseteq \mathscr{C}(M)$. Further, assume M satisfies the following conditions*

(a) *for any $A_1, A_2 \in \mathscr{A}$, if $A_1 \overset{+\mathscr{A}}{\vdash} A_1, A_2$, then $A_2 \overset{+\mathscr{A}}{\vdash} A_1$;*

(b) *for any $A \in \mathscr{A} \cup \mathscr{I}(M)$, any $B \notin \mathscr{A}$, and any C_1, C_2, C_3, if*

$$A \overset{*\mathscr{A}}{\vdash} C_1 \overset{*\mathscr{A}}{\dashv} C_2 \dashv B \vdash C_3,$$

then $C_2 = C_3$;

(c) *for any $A_1 \in \mathscr{A} \cup \mathscr{I}(M)$, any $A_2 \in \mathscr{A}$, and any B, if $A_1 \overset{*\mathscr{A}}{\vdash} B \overset{*\mathscr{A}}{\dashv} A_2$, then $A_1 = A_2$.*

Then $\overline{M\#}$ accepts the same language as M within the same space bound. □

3. Simulation of Space-Bounded Symmetric Turing Machines by Symmetric Counter Machines without Input Terminals

In proving that UWP(6) is SSPACE(n)-hard, we will show (1) how a linear-space-bounded symmetric Turing machine can be simulated by a specific exponential-space-bounded symmetric counter machine with only 3 counters

and without input terminal, and (2) how to describe the computations of this specific symmetric counter machine as instance of UWP(6). This section is devoted to (1), whereas (2) will be done in the next section.

Since we are interested in linear-space-bounded symmetric Turing machines, we may assume, by Theorem 2 (linear speed-up) and Theorem 3 (tape compression) in [9], and Lemma 2.1, that

(1) our linear-space bounded symmetric Turing machines have only 1 tape;

(2) the input alphabet includes two special symbols % and \$, the left and right endmarkers, respectively;

(3) the read/write head never moves left from % nor right from \$; (Initially, the input string appears between % and \$ on the tape and % is scanned. Further, % and \$ are never rewritten.)

(4) in an accepting configuration, all tape cells contain the blank symbol # except the two cells containing the endmarkers %, \$, and the head scans the left endmarker.

Before introducing the specific model of symmetric 3-counter machines without input terminals, we first give the definition of counter machines, which is slightly different from [4]. A k-counter machine has k counters and one read-only input terminal holding $¢w\&$, where w is the input string and $¢$, & are special symbols. Each counter is a semi-infinite tape that is infinite to the right.

Definition 3.1. A k-counter machine (k-CM for short) is a system

$$C=(k, Q, \Sigma, \Delta, q_0, F, ¢, \&, \#, \#'),$$

where

k is the number of counters,
Q is the finite set of states,
Σ is the input alphabet,
$q_0 \in Q$ is the initial state,
$F \subseteq Q$ is the set of final states,
$\#$ is the blank symbol, $\#'$ is the pseudo-blank symbol,
$¢$ and & are special symbols not in $\Sigma \cup \{\#\}$,
Δ is the set of transitions of the form

$$t=(p, t_0, t_1, \ldots, t_k, q), \quad p, q \in Q,$$

where

- t_0 is either[3] (ab, D) with $ab \in ¢\Sigma \cup \Sigma\& \cup \Sigma^2$ and $D \in \{-1, +1\}$, or (a, o), $a \in \Sigma \cup \{¢, \&\}$,
- for $j=1, \ldots, k$, t_j is in one of the following forms:

$$(\&\#, \&\&), \quad (\&\#', \&\&), \quad (\&\&, \&\#'), \quad (\&, \&), \quad (¢, ¢), \quad (¢\#, ¢\&),$$
$$(¢\#', ¢\&), \quad (¢\&, ¢\#'), \quad (\&\&, \&\#) \text{ and } (¢\&, ¢\#).$$

(*Remark.* Note that we introduce the pseudo-blank symbol for technical convenience only. This is done, since we will apply Lemma 2.1 for counter machines.)

[3] The read-only input head cannot move left (right) from the symbol ¢(&)

Initially, C is in state q_0, the read-only input terminal contains $\mathfrak{c}w\&$ where $w\in\Sigma^*$ is the input string, and all counters contain only the endmarker in their leftmost cells. The counters can hold only the symbols &, #′ apart from $\mathfrak{c}$ in the left most cell.

A transition $t=(p, t_0, t_1, \ldots, t_k, q)$ defines the action of C at some step in a computation. t_0 corresponds to an action on the input terminal. If $t_0=(a, o)$, then C scanning a does not move the input head. If $t_0=(ab, (-)1)$, then C scanning $a(b)$ moves to the right (left).

t_j, $j=1, \ldots, k$, defines the action on the jth counter. If, for instance, $t_j=(\&\#, \&\&)$, then counter j is incremented. $(\&\&, \&\#')$ means a decrement, while $(\&, \&)$ means that no action is performed. Other possibilities are interpreted in an analogous way.

A configuration of C is

$$(q, w, i, d_1, \ldots, d_k),$$

where

(1) $q\in Q$,
(2) w is the input string, $|w|=n$,
(3) $0\leqq i\leqq n+1$ is the position of the input head,
(4) d_j, $1\leqq j\leqq k$, is the total number of &'s in counter j.

The "yield" relation $\xrightarrow[C]{}$ between configurations of C can be defined as usual. $\xrightarrow[C]{*}$ denotes the reflexive and transitive closure of $\xrightarrow[C]{}$. We often write $\rightarrow$ and $\xrightarrow{*}$ if C is understood.

The language accepted by C is defined as

$$L(C):=\{w\in\Sigma^* \mid (q_0, w, 0, 0, \ldots, 0)\xrightarrow{*}(q_f, w, i, d_1, \ldots, d_k) \text{ where } q_f\in F,\ 0\leqq i\leqq |w|+1,\ d_1, \ldots, d_k\in\mathbb{N}\}.$$

In order to obtain symmetric CM's we define "reversed" transitions as in the case of Turing machines.

Definition 3.2. Let

$$C=(k, Q, Z, \Delta, q_0, F, \mathfrak{c}, \&, \#, \#')$$

be a k-CM. The symmetric closure of C is

$$\bar{C}=(k, Q, \Sigma, \Delta\cup\Delta^{-1}, q_0, F, \mathfrak{c}, \&, \#, \#')$$

where $\Delta^{-1}:=\{t^{-1} \mid t\in\Delta\}$. C is called a *symmetric* k-CM iff $C=\bar{C}$.

A k-CM operates in space $S(n)$, if starting with input w, $|w|=n$, it requires at most $S(n)$ tape cells.

If C is symmetric, the yield relation is denoted by $\overset{*}{\underset{c}{\leftrightarrow}}$, or by $\overset{*}{\leftrightarrow}$ if C is understood.

Remark 3.3. According to our definition, a k-CM is a special TM. Thus we may apply Lemma 2.1 when we want to obtain the symmetric closure of a CM. Note that, by definition, the passage from a CM C to $C\#$ (i.e. putting C into "normal form") can be done without introducing a "pseudoblank" symbol.

We now show how to simulate linear-space-bounded symmetric Turing machines by exponential-space-bounded symmetric 3-CM's without input terminals.

Let M be a fixed linear-space bounded symmetric Turing machine with input alphabet Σ containing two special symbols $\%_0$, \$ and tape alphabet $\Gamma \supseteq \Sigma$. Let card (Γ) be $l-1$ and g be a bijection from Γ onto $\{1, 2, \ldots, l-1\}$. (We assume that M satisfies the four conditions stated at the beginning.) The 3-CM C without input terminal is constructed as follows.

The tape content

$$\%_0 a_r a_{r-1} \ldots a_0 a b_0 \ldots b_s \$$$

of M (scanning a) is encoded by counter 1 and counter 2 of C:

- counter 1 holds $g(\$)\, l^{s+1} + g(b_s)\, l^s + g(b_{s-1})\, l^{s-1} + \ldots + g(b_0)$,
- counter 2 holds $g(\%_0)\, l^{r+1} + g(a_r)\, l^r + g(a_{r-1})\, l^{r-1} + \ldots + g(a_0)$,

where $r+s+3=n$, and a_0, a, b_0 are stored in the finite control of C. Thus the left (right) portion is encoded by counter 2(1).

If $w = w_1 \ldots w_n$ is the input string of M, then C, in its initial configuration, contains:

- $g(w_1) + \ldots + g(w_{n-1})\, l^{n-2} + g(w_n)\, l^{n-1} + g(\$)\, l^n = \tilde{w}$ in counter 1,
- 0 in counter 2,
- 0 in counter 3,

and $\%_0$ is stored in the finite control (since M is scanning $\%_0$).

C simulates each move of M by a sequence of moves. (The simulation is similar to that in the proof of Theorem 3.1 in [4].) First, C chooses the transition of M to be simulated. If, for example, the transition changes ab_0 to cd and moves the head one cell right, then C performs the following steps: (A state of C consists of a state that is either a state of M or some particular state, and some other information.)

- Multiply the content of counter 2 by l and add $g(c)$;
- Divide the content of counter 1 by l twice to obtain the remainder $g(b_1)$, and store cdb_1 in the finite control.
- Multiply the content of counter 1 by l and add $g(b_1)$, empty counter 3, and enter the next state (determined by the transition chosen at the beginning of the simulation).

The initial state of C is q_0, and its initial configuration is $(q_0, \tilde{w}, 0, 0)$. In its initial state, C, scanning $\%_0$, will determine w_1 and store $\#$, $\%_0$, w_1 in its finite control before entering the initial state of M.

The final state of C is q_f: as soon as C enters a final state of M, it makes a transition to state q_f. Thus, the final configuration of C is[4] $(q_f, \tilde{v}, 0, 0)$, where $\tilde{v} := g(\#) + g(\#)\, l + \ldots + g(\#)\, l^{n-1} + g(\$)\, l^n$.

---

[4] Recall that an accepting configuration of M has the property that every cell contains the blank symbol, except the two cells containing the endmarkers

Lemma 3.4. *As above, let M be a linear-space-bounded symmetric Turing machine and C be the* 3-CM *without input terminal constructed above. Let w be an input string to M. Then w is accepted by M iff*

$$(q_0, \tilde{w}, 0, 0) \overset{*}{\underset{\bar{C}}{\leftrightarrow}} (q_f, \tilde{v}, 0, 0).$$

Further, $\bar{C}$ is c^n-space-bounded for some fixed constant c.

Proof. Note that in the construction of C from M no t_j of the form $(c\&, \not{c}\#)$ or $(\&\&, \&\#)$ is needed. Hence, C can be taken to fulfill the role of $C\#$ and Lemma 2.1 can be applied. We now show that the conditions (a), (b) and (c) of Lemma 2.1 are satisfied by C. Let $\mathscr{A}$ be the set of configurations of C that contain some state of M. (a) is satisfied, since M is symmetric. (b) follows from the fact that C, in a state that does not contain a state of M, is deterministic, and (c) follows from the fact that each simulation of a transition of M happens deterministically. □

4. UWP(k), $k \geqq 6$, is SSPACE(n)-hard

In this section we show that UWP(k), $k \geqq 6$, is SSPACE(n)-hard. In doing so we reduce the computations of the symmetric 3-CM without input terminal constructed in Sect. 3 to UWP(6). Consider the symmetric 3-CM $\bar{C}$ without input terminal constructed in Sect. 3. $\bar{C}$ simulates the linear-space-bounded symmetric Turing machine M. Let w be an input string of M. Then, by Lemma 3.4, w is accepted by M iff

$$(q_0, \tilde{w}, 0, 0) \overset{*}{\underset{\bar{C}}{\leftrightarrow}} (q_f, \tilde{v}, 0, 0),$$

where $\tilde{v}$ is defined as in Sect. 3. The proof idea is to encode the counters of $\bar{C}$, each by a pair of generating symbols. Since $\bar{C}$ has a fixed number of states, encoding them can be done easily. A transition of $\bar{C}$ will be simulated by a relation of the presentation to be constructed.

Proposition 4.1. UWP(k), $k \geqq 6$, *is* SSPACE(n)*-hard.*

Proof. We construct the presentation (S, R) as follows. S consists of the symbols $s_1, \ldots, s_6$. Counter j, $j = 1, 2, 3$, is encoded by the pair s_{2j-1}, s_{2j}. The relations in R will simulate the transitions of $\bar{C}$.

Since $\bar{C}$ operates in space c^n for some fixed constant c, the content of each counter does not exceed c^n. Let max denote the number $c^n + 1$. If counter j, $j = 2, 3$, contains d_j, then it is coded by $s_{2j-1}^{d_j} s_{2j}^{\max - d_j}$. Hence the two counters 2, 3 of $\bar{C}$ are coded together by the word

$$s_3^{d_2} s_4^{\max - d_2} s_5^{d_3} s_6^{\max - d_3}.$$

Now let $\mathrm{Card}(Q) = m$ and $Q = \{q_0, q_1, \ldots, q_{m-1}\}$. W.l.o.g. we assume that $q_f = q_{m-1}$. If $\bar{C}$ is in state q_i, then q_i is encoded by the symbols s_1, s_2, i.e. if counter 1 contains d_1, then q_i and the content of counter 1 are encoded by $s_1^{i \cdot \max + d_1} s_2^{(m-i) \cdot \max - d_1}$.

Thus, a configuration of $\bar{C}$ of the form (q_i, d_1, d_2, d_3) is encoded by the word

$$s_1^{i\cdot \max + d_1}\, s_2^{(m-i)\max - d_1}\, s_3^{d_2}\, s_4^{\max - d_2}\, s_5^{d_3}\, s_6^{\max - d_3}$$

in a unique way.

The transitions of $\bar{C}$ are simulated by the relations of R as follows. A transition of the form

$$(q_i, (\not c\#, \not c\&), (\&\#, \#\#), (\&\#, \#\#), q_j)$$

corresponds to the relation

$$s_1^{i\cdot \max}\, s_2^{(m-i)\max + 1}\, s_3\, s_5 = s_1^{j\cdot \max + 1}\, s_2^{(m-j)\max}\, s_4\, s_6,$$

and a transition of the form

$$(q_i, (\&\#, \#\#), (\&, \&), (\&\#, \#\#), q_j)$$

corresponds to the relation

$$s_1^{i\cdot \max + 1}\, s_2^{(m-i)\max}\, s_3\, s_5 = s_1^{j\cdot \max}\, s_2^{(m-j)\max + 1}\, s_3\, s_6.$$

Other forms of transitions are treated in an analogous way.

We proceed to show that our construction works. Let R denote the set of defining relations defined as above. For $j=1, 2, 3$ let $e_j(x)$ denote the sum of the exponents of s_{2j-1} and s_{2j} on $x \in S^{\oplus}$. From the construction we see that $x \sim y \bmod R$ implies $e_j(x) = e_j(y)$, $j=1, 2, 3$.

Now the initial configuration of $\bar{C}$ with input string w, $(q_0, \tilde{w}, 0, 0)$, corresponds to the word $s_1^{\tilde{w}}\, s_2^{m\cdot \max - \tilde{w}}\, s_3^0\, s_4^{\max}\, s_5^0\, s_6^{\max}$, and the final configuration of C, $(q_{m-1}, \tilde{v}, 0, 0)$ corresponds to the word

$$s_1^{(m-1)\cdot \max + \tilde{v}}\, s_2^{\max - \tilde{v}}\, s_3^0\, s_4^{\max}\, s_5^0\, s_6^{\max}.$$

We show

Claim. $(q_0, \tilde{w}, 0, 0) \overset{*}{\underset{\bar{C}}{\leftrightarrow}} (q_{m-1}, \tilde{v}, 0, 0)$ iff

$$s_1^{\tilde{w}}\, s_2^{m\cdot \max - \tilde{w}}\, s_3^0\, s_4^{\max}\, s_5^0\, s_6^{\max} \sim s_1^{(m-1)\max + \tilde{v}}\, s_2^{\max - \tilde{v}}\, s_3^0\, s_4^{\max}\, s_5^0\, s_6^{\max} \bmod R.$$

Proof of Claim. The only-if part is straightforward. We show the if part. Let

$$x = s_1^{\tilde{w}}\, s_2^{m\cdot \max - \tilde{w}}\, s_4^{\max}\, s_6^{\max}$$

and

$$y = s_1^{(m-1)\max + \tilde{v}}\, s_2^{\max - \tilde{v}}\, s_4^{\max}\, s_6^{\max}.$$

$x \sim y \pmod R$ implies that there is a derivation

$$D\colon x = u_0 \leftrightarrow u_1 \leftrightarrow \ldots \leftrightarrow u_l = y.$$

We may assume that D contains no loops, i.e. $u_\lambda \neq u_{\lambda'}$ for $0 \leqq \lambda \neq \lambda' \leqq l$.

Observe that $e_1(u_\lambda) = m\cdot \max$, $e_2(u_\lambda) = e_3(u_\lambda) = \max$.

Clearly, each u_λ, $0 \leqq \lambda \leqq l$, represents a configuration, say K_λ, of $\bar{C}$ and for $\lambda = 0, \ldots, l-1$

$$u_\lambda \underset{R}{\leftrightarrows} u_{\lambda+1} \quad \text{iff} \quad K_\lambda \underset{\bar{C}}{\leftrightarrows} K_{\lambda+1}.$$

Therefore, $(q_0, \tilde{w}, 0, 0) = K_0 \overset{*}{\underset{\bar{C}}{\leftrightarrows}} K_l = (q_{m-1}, \tilde{v}, 0, 0)$.
This completes the proof of the claim. □

To complete the proof of Proposition 4.1, it remains to show that given w the instance (S, R), x, y of UWP(6) defined above can be constructed by a log-space-bounded Turing transducer. First, note that given w, the set of transitions of $\bar{C}$ and $\tilde{w}$, $\tilde{v}$ can be computed by a log-space-bounded transducer: the transitions of $\bar{C}$ are computed using the information about the transitions of M (which is fixed) and $\tilde{w}$, $\tilde{v}$ can be computed in a straightforward way. Second, given the set of transitions of $\bar{C}$ and $\tilde{w}$, $\tilde{v}$, the instance (S, R), x, y of UWP(6), as defined above, can be computed by a log-space-bounded Turing transducer. These two log-space-bounded Turing transducers can be, by well known techniques, composed so that the resulting machine is a log-space-bounded Turing transducer that with input w outputs (S, R), x, y. Thus, UWP(6) is SSPACE(n)-hard. This completes the proof of Proposition 4.1. □

5. UWP(k) is Solvable in Symmetric Linear Space

In this section we show that UWP(k) is solvable in SSPACE(n), which is a consequence of the fact that the general word problem for commutative semigroups is solvable in exponential space. This is announced in [1] and formally shown in [10]. We reproduce some of their results in order to make this paper self-contained.

In the proof that the general word problem for commutative semigroups is in exponential space, a connection between this and the membership problem for polynomial ideals was used. Let $\mathbb{Q}[S] = \mathbb{Q}[s_1, \ldots, s_k]$ denote the ring of polynomials in the variables $s_1, \ldots, s_k$ with rational coefficients. Consider the set of defining relations $R = \{(u_1, v_1), \ldots, (u_m, v_m)\}$. Let $I(R)$ denote the ideal

$$I(R) = \langle u_1 - v_1, \ldots, u_m - v_m \rangle \subseteq \mathbb{Q}[s_1, \ldots, s_k].$$

It holds that for $x, y \in S^\oplus$:

$$x \sim y \pmod{R} \quad \text{iff } x - y \in I(R).$$

(For a proof see for instance [3], p. 187.)

For $z = s_1^{e_1} \ldots s_k^{e_k}$ let $h(z) := e_1 + \ldots + e_k$ denote the exponent sum of z. Further, for a finite subset $Z \subseteq S^\oplus$ let $h(Z)$ denote the number $\max\{h(z) \mid z \in Z\}$. We need the following facts, whose proofs can be found in [10].

Fact 5.1. If $x - y \in I(R)$, then there are polynomials $p_1, \ldots, p_m \in \mathbb{Q}[S]$ such that

(1) $x - y = \sum_{i=1}^{m} p_i(u_i - v_i)$,

(2) $\deg(p_i) \leqq h(\{x, y\}) + (m h(R))^{2^k}$

for all $i=1, \ldots, m$, where $\deg(p)$ denotes the degree of the polynomial p and $h(R) := h(\{u_i, v_i \mid i=1, \ldots, m\})$.

Fact 5.2. If $x-y=\sum_{i=1}^{m} p_i(u_i-v_i)$, then there is a derivation

$$x=z_0 \leftrightarrow z_1 \leftrightarrow \ldots z_l=y$$

in (S, R) such that for $\lambda=0, 1, \ldots, l$

$$h(z_\lambda) \leqq \max\{\deg(v_i p_i) \mid 1 \leqq i \leqq m\}.$$

From Facts 5.1 and 5.2 we have

Lemma 5.3. $x \sim y \pmod R$ *iff there is a derivation*

$$x=z_0 \leftrightarrow z_1 \leftrightarrow z_2 \ldots z_l=y$$

in (S, R) *such that for* $\lambda=0, 1, \ldots, l$

$$h(z_\lambda) \leqq h(R)[h(\{x, y\})+(mh(R))^{2^k}]. \quad \square$$

Proposition 5.4. UWP(k) *is in* SSPACE(n).

Proof. The proof is left to the reader. $\square$

6. The Complexity of UWP(1)

In this section we show that UWP(1) is solvable in polynomial time. Let $S=\{s\}$ and $R=\{(u_1, v_1), \ldots, (u_m, v_m)\} \subseteq S^\oplus \times S^\oplus$. For $x \in S^\oplus$ let $[x]$ denote the equivalence class $[x]=\{z \in S^\oplus \mid z \sim x \pmod R\}$. W.l.o.g. we assume $u_i \neq v_i$.

Let $\leqq$ denote the following partial order on $S^\oplus$:

$$s^e \leqq s^f \quad \text{iff} \quad e \leqq f.$$

$S^\oplus$ is linearly ordered with respect to $\leqq$ and every subset of $S^\oplus$ has exactly one minimal element. Let $\operatorname{Min}[x]$ denote the minimal element of $[x]$. Further let $d \in \mathbb{N}$ denote $gcd(d_1, \ldots, d_m)$, where $d_i=e_i-f_i$ and $u_i=s^{e_i}$, $v_i=s^{f_i}$.

Lemma 6.1. *Either* $[x]=\{x\}$ *or* $[x]=\operatorname{Min}[x]+\{s^{\lambda d} \mid \lambda \in \mathbb{N}\}$.

Proof. If no relation in R is applicable, then $[x]=\{x\}$. Otherwise, $[x]$ is obviously infinite. In this case, $y \sim y+s^d$ for every $y \in [x]$. To see this consider d. Since $d=gcd(d_1, \ldots, d_m)$, there are integers $r_1, r_2, \ldots, r_m$ and that

$$d=\sum_{i=1}^{m} r_i d_i.$$

Let y be large enough such that we can apply $|r_i|$ times the relation (u_i, v_i) or (v_i, u_i) successively for $i=1, \ldots, m$. The last word of this derivation is $y+s^d$. Since we have a congruence, this situation holds for every $y \in [x]$. Thus $\operatorname{Min}[x]+\{s^{\lambda d} \mid \lambda \in \mathbb{N}\} \subseteq [x]$.

The fact that $[x] \subseteq \operatorname{Min}[x]+\{s^{\lambda d} \mid \lambda \in \mathbb{N}\}$ is straightforward. $\square$

Given x, y and R we can check whether $x \sim y \bmod R$ in polynomial time as follows: check whether any $r \in R$ is applicable at x. If not, $[x] = \{x\}$ and $y \nsim x \bmod R$. Otherwise, check if any relation $r \in R$ is applicable at y. If not, then $[y] = \{y\}$ and $y \nsim x \bmod R$. Otherwise $[x]$ and $[y]$ are infinite. In this case check whether $d \mid e - f$, where $x = s^e$, $y = s^f$. If it is so, then $x \sim y \bmod R$. Otherwise $x \nsim y \bmod R$. Since all these steps can be carried out in polynomial time, we have

Proposition 6.2. UWP(1) *is solvable in polynomial time.* □

Open Problem. What is the exact complexity of UWP(k), $2 \leqq k \leqq 5$?

Remark. Recently, in connection with certain subclasses of polynomial ideals, we have been able to show in [6] that UWP(4) is NP-hard. The reduction is technically complicated and does not apply to the cases of lower dimensions. It seems that some more insights into the structure of congruences on $\mathbb{N}^k$ are needed in order to obtain tight bounds for the above subcases.

Acknowledgement. The author thanks the referees for many valuable suggestions for improving the presentation of this paper.

References

1. Cardoza, E., Lipton, R., Meyer, A.R.: Exponential Space Complete Problems for Petri Nets and Commutative Semigroups: Preliminary Report, Proc. 8th Ann. ACM Symposium on Theory of Computing, pp. 50–54, 1976
2. Biryukov, A.P.: Some Algorithmic Problems for Finitely Defined Commutative Semigroups. Sib. Math. J. **8**, 384–391 (1967)
3. Eilenberg, S., Schützenberger, M.P.: Rational Sets in Commutative Monoids. J. Algebra **13**, 173–191 (1969)
4. Fisher, P.C., Meyer, A.R., Rosenberg, A.L.: Counter Machines and Counter Languages. Math. Syst. Theory **2**, 265–283 (1968)
5. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Reading, MA: Addison-Wesley 1979
6. Huynh, D.T.: The Complexity of the Membership Problem for Two Subclasses of Polynomial Ideals. Manuscript, 1984 (To appear in SIAM J. Comput.)
7. Huynh, D.T.: The Complexity of the Equivalence Problem for Commutative Semigroups and Symmetric Vector Addition Systems. Proc. 17th STOC, pp. 405–412, 1985
8. Van Leeuwen, J.: A Partial Solution to the Reachability Problem for Vector Addition Systems. Proc. 6th Ann. ACM Symposium on Theory of Computing, pp. 303–307, 1974
9. Lewis, H.R., Papadimitriou, C.H.: Symmetric Space-Bounded Computation. Theor. Comput. Sci. **19**, 161–187 (1982)
10. Mayr, E., Meyer, A.R.: The Complexity of the Word Problems for Commutative Semigroups and Polynomial Ideals. Adv. Math. **45**, 305–329 (1982)
11. Mayr, E.: An Algorithm for the General Petri Net Reachability Problem. Proc. 13th Ann. Symposium on Theory of Computing, pp. 238–246, 1981
12. Taiclin, M.A.: Algorithmic Problem for Commutative Semigroups. Sov. Math. Dokl. **9**, 201–204 (1968)

Received March 15, 1984/May 29, 1985