On Two-Way FA with Monotonic Counters and Quadratic Diophantine Equations

Oscar H. Ibarra[1] ⋆ and Zhe Dang[2]

[1]Department of Computer Science
University of California
Santa Barbara, CA 93106, USA

[2]School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164, USA

Abstract. We show an interesting connection between two-way deterministic finite automata with monotonic counters and quadratic Diophantine equations. The automaton M operates on inputs of the form $a_1^{i_1} \cdots a_n^{i_n}$ for some fixed n and distinct symbols $a_1, \cdots, a_n$, where $i_1, \cdots, i_n$ are nonnegative integers. We consider the following reachability problem: Given a machine M, a state q, and a Presburger relation E over counter values, is there $(i_1, \cdots, i_n)$ such that M, when started in its initial state on the left end of the input $a_1^{i_1} \cdots a_n^{i_n}$ with all counters initially zero, reaches some configuration where the state is q and the counter values satisfy E? In particular, we look at the case when the relation E is an equality relation, i.e., a conjunction of relations of the form $C_i = C_j$. We show that this case and variations of it are equivalent to the solvability of some special classes of systems of quadratic Diophantine equations. We also study the nondeterministic version of two-way finite automata augmented with monotonic counters with respect to the reachability problem. Finally, we introduce a technique which uses decidability and undecidability results to show "separation" between language classes.

Key Words: two-way finite automaton, monotonic counter, reachability problem, quadratic Diophantine equation, Presburger relation, semilinear set, decidability.

1 Introduction

Two-way finite automata are not stronger than one-way finite automata in terms of language acceptance, since they both accept regular languages. However, when equipped with additional unbounded storage devices, these two classes of automata could be completely different (in computing power). A counter is one such device. It can store an integer number and can be incremented/decremented by one and tested against zero. It is well known that the emptiness problem for one-way finite automata augmented with a counter is decidable (in fact, emptiness is decidable even when the counter is replaced by a pushdown stack). However, from [9], the emptiness problem for two-way finite automata augmented with one counter is undecidable, even when the input is unary. It is interesting to study under what restrictions of the counter the problem becomes decidable. One such restriction is to require that the counter be reversal-bounded (i.e., the number of alternations between nondecreasing mode and nonincreasing mode and vice-versa is bounded by a fixed integer independent of the computation) [5]. We use 2DCM(1) (resp. 2NCM(1)) to denote a deterministic (resp. nondeterministic) two-way finite automaton augmented with one reversal-bounded counter. It has been found useful to restrict the inputs to be from a bounded language (i.e., the inputs are in the form $a_1^{i_1} \ldots a_n^{i_n}$ for some fixed n and distinct symbols $a_1, \ldots, a_n$, where $i_1, \ldots, i_n$ are nonnegative integers) [5]. It is known that the emptiness problem for 2DCM(1) over a bounded language is decidable [4]. This result was later generalized to 2DCM(1) (not necessarily over a bounded language) [6]. For the nondeterministic counterpart,

⋆ The work by Oscar H. Ibarra has been supported in part by NSF Grants IIS-0101134 and CCR02-08595.

it has been recently shown that the emptiness problem for 2NCM(1) over a bounded language is decidable [1]. However, in general, the emptiness problem for 2NCM(1) is still an open problem.

Clearly, it is also desirable to study two-way finite automata augmented with multiple reversal-bounded counters. However, it turns out that these automata do not have a decidable emptiness problem even when they are deterministic and over a bounded language [5]. In this paper, we study these automata where the counters are monotonic (i.e., nondecreasing) instead of reversal-bounded. More precisely, we consider two-way deterministic finite automata M augmented with monotonic counters over a bounded language. On each transition, M is able to read an input symbol and increment some counters by 1. On an input, M either eventually enters a deadlock state when no further transition is possible (i.e., M halts) or runs forever. Thus, we do not assume that M always halts on all the inputs. Notice that the monotonic counters do not participate in the dynamics of M. Clearly, the counters are quite useful, e.g., in counting the number of time some particular event occurs (i.e., the number of reads for a symbol).

We are interested in the following reachability problem: Given a machine M, a state q, and a Presburger relation E over counter values, is there $(i_1, \ldots, i_n)$ such that M, when started in its initial state on the left end of the input $a_1^{i_1} \ldots a_n^{i_n}$ with all counters initially zero, reaches some configuration where the state is q and the counter values satisfy E? For notational convenience, we sometimes write E_q to mean that the relation E must be satisfied at state q, and refer to E_q as the relation. One can show that when M always halts, the problem is decidable. However, when M does not always halt, the problem is open, even for the special case when E is an equality relation, i.e., a conjunction of relations of the form $C_i = C_j$ (where C_i and C_j are counters). We show, however, that this special case is equivalent to the solvability (which is still unknown) of a special class of systems of quadratic Diophantine equations. We also study a number of variations of the problem and their decidability. For example, we show that given M and equality relations $E_{q_1}, \ldots, E_{q_m}$, the following problem is undecidable: is there $(i_1, \ldots, i_n)$ such that M has a computation where the counter values satisfy each E_{q_i} at some time during the computation (not necessarily at the same time, and not necessarily in the given order)? The undecidability holds even when each E_{q_i} has only two conjuncted terms. However, when each E_{q_i} has only one term, the problem is decidable. Again, both of these cases are equivalent to solving special classes of quadratic Diophantine equations. We also study the nondeterministic version of two-way finite automata augmented with monotonic counters and show that (in contrast to the deterministic model) the reachability problem is undecidable when E is an equality relation, or when E is a semi-equality relation where each E_{q_i} a single term.

We note that the model and techniques presented in this paper are incomparable to those investigated in [4] (though the title looks similar). The paper [4] focuses on 2DCM(1) over bounded languages and uses Lipshitz's theorem [7], while here, the model deals with multiple counters with Presburger constraints. However, interestingly, we are able to use the results obtained in this paper to prove some new results on 2DCM(1) and 2NCM(1). It was not known whether the class of languages accepted by 2NCM(1) is the same as the class of languages accepted by 2DCM(1). That is, as language acceptors, are 2NCM(1)'s strictly more powerful than 2DCM(1)'s? We show how our decidability and undecidability results can be used to answer this question affirmatively: there is a language that can be accepted by a 2NCM(1) but not by a 2DCM(1). The technique is interesting as it does not use the usual pumping lemma, crossing sequence, or other combinatorial arguments.

The rest of the paper is organized as follows. In Section 2, we give some known results on reversal-bounded counters and semilinear sets needed in the paper. In Section 3, we define the model of a deterministic finite automaton with monotonic counters and the fundamental problem concerning this machine. In Section 4, we show the connections between the model and its variations to some classes of quadratic Diophantine equations. In Section 5, we discuss the nondeterministic model and present some further results. In Section 6, we look at the the language recognition power of 2DCM(1) and 2NCM(1). Section 7 is a brief conclusion.

2 Preliminaries

Let k be a nonnegative integer. A k-counter machine is a two-way nondeterministic finite automaton with input endmarkers (two-way NFA) augmented with k counters, each of which can be incremented by 1, decre-

mented by 1, and tested for zero. We assume, w.l.o.g., that each counter can only store a nonnegative integer, since the sign can be stored in the states. If r is a nonnegative integer, let 2NCM(k,r) denote the class of k-counter machines where each counter is r *reversal-bounded*, i.e., it makes at most r alternations between nondecreasing and nonincreasing modes in any computation; e.g., a counter whose values change according to the pattern 0 1 1 2 3 4 $\underline{4\ 3}$ 2 1 $\underline{0\ 1}$ $\underline{1\ 0}$ is 3-reversal, where the reversals are underlined. For convenience, we sometimes refer to a machine in the class as a 2NCM(k,r). We are interested, in particular, with 2NCM(k,0), i.e., the counters are monotonic (i.e., nondecreasing). Clearly, we may assume that the counters in a 2NCM(k,0) do not participate in the dynamic of the machine, since the finite-state control need only keep track of when the counters become positive.

A 2NCM(k,r) is *finite-crossing* [3, 5] if there is a positive integer d such that in any computation, the input head crosses the boundary between any two adjacent cells of the input no more than d times. Note that a 1-crossing 2NCM(k,r) is a one-way nondeterministic finite automaton augmented with k r-reversal counters. 2NCM(k) will denote the union of 2NCM(k,r), $r = 1, 2, \ldots$. For deterministic machines, we use 'D' in place of 'N'. If M is a machine, $L(M)$ denotes the language it accepts. A language is *bounded* if it is a subset of $a_1^* a_2^* \ldots a_n^*$ for some fixed n and distinct symbols symbols $a_1, a_2, \ldots, a_n$.

The following theorems summarize the important results concerning reversal-bounded counter machines which we will need in the paper.

Theorem 1. *There is a* fixed r *such that the emptiness problem for 2DCM(2,r) over bounded languages is undecidable [5].*

Theorem 2. *The emptiness problem is decidable for the following classes:*
(a) 2DCM(1) [6].
(b) 2NCM(1) over bounded languages [1].
(c) 2NCM(k) over a unary alphabet, for every k [6].
(d) finite-crossing 2NCM(k), for every k [3, 5].

Let Y be a finite set of variables over integers. For all integers a_y, with $y \in Y$, b and c (with $b > 0$), $\sum_{y \in Y} a_y y < c$ is an *atomic linear relation* on Y and $\sum_{y \in Y} a_y y \equiv_b c$ is a *linear congruence* on Y. A *linear relation* on Y is a Boolean combination (using $\neg$ and $\wedge$) of atomic linear relations on Y. A *Presburger formula* on Y is the Boolean combination of atomic linear relations on Y and linear congruences on Y. A set P of tuples of nonnegative integers is *Presburger-definable* or a *Presburger relation* if there exists a Presburger formula $\mathcal{F}$ on Y such that P is exactly the set of the solutions for Y that make $\mathcal{F}$ true. It is well known that Presburger formulas are closed under quantification.

Let $\mathbf{N}$ be the set of nonnegative integers and n be a positive integer. A subset S of $\mathbf{N}^n$ is a *linear set* if there exist vectors $v_0, v_1, \ldots, v_t$ in $\mathbf{N}^n$ such that $S = \{v \mid v = v_0 + a_1 v_1 + \ldots + a_t v_t, \forall 1 \leq i \leq t, a_i \in \mathbf{N}\}$. S is a *semilinear set* if it is a finite union of linear sets. It is known that S is a semilinear set if and only if S is Presburger-definable [2].

Let Σ be an alphabet consisting of n symbols $a_1, \ldots, a_n$. For each string (word) w in Σ^*, we define the *Parikh map* of w, denoted by $p(w)$, as follows:

$$p(w) = (i_1, \ldots, i_n), \text{ where } i_j \text{ is the number of occurrences of } a_j \text{ in } w.$$

If L is a subset of Σ^*, the *Parikh map* of L is defined by $p(L) = \{p(w) \mid w \in L\}$. L is a *semilinear language* if its Parikh map $p(L)$ is a semilinear set. We will need the following theorem from [3, 5]:

Theorem 3. *Let M be a finite-crossing 2NCM(c). Then $p(L(M))$ is a semilinear set effectively computable from M.*

Note that the result above is not true for machines that are not finite-crossing. For example, a 2DCM(1,1) can recognize the language $\{0^i 1^j \mid i \text{ divides } j\}$, which is not semilinear. The following theorem can be easily verified.

Theorem 4. *Let S be a subset of $\mathbf{N}^n$ and $L = \{a_1^{i_1} \ldots a_n^{i_n} \mid (i_1, \ldots, i_n) \in S\}$. If S is Presburger-definable (i.e., semilinear), then:*

(a) L can be accepted by a 1-crossing 2DCM(k) for some k.

(b) L can be accepted by a 2DCM(1).

3 The Fundamental Problem

Consider a 2DCM(k,0) M over a bounded language. Thus, M is a *deterministic* two-way finite automaton augmented with k monotonic counters $C_1, \ldots, C_k$. The two-way input, w (which is provided with left and right endmarkers), comes from a bounded language, i.e., $w = a_1^{i_1} \ldots a_n^{i_n}$ for some fixed n and distinct symbols $a_1, \ldots, a_n$, and $i_1, \ldots, i_n$ are nonnegative integers. The counters are initially zero and can be incremented by 0 or 1 at each step, but cannot be decremented. They do not participate in the dynamic of the machine. For convenience, we shall simply call M a 2FAMC. Note that we do not assume that the machine halts on all inputs..

It is interesting to note that the set of tuples of nonnegative integers "generated" by a 2FACM (at a specified state) need not be semilinear (Presburger) in general. For example, consider a 2FACM with two monotonic counters C_1 and C_2. On unary input x of length n, M initially stores n in C_1. Then M makes left-to-right and right-to-left sweeps of the input, adding n to C_2 after every left-to-right sweep. M iterates this process without halting. Let s be the state of M just after a left-to-right sweep. Then the set of tuples of values of the counters when it is in state s is $Q_s = \{(n, kn) \mid n \geq 0, k > 0\}$, which is not semilinear.

We are interested in the problem of deciding, given a 2FAMC M and a Presburger relation E, whether the set of tuples generated by M satisfies E. It turns out that even for simple relations, the decidability of this problem is open.

An atomic equality relation on the counters is a relation of the form $C_i = C_j$, $i \neq j$. An *equality relation* E is a conjunction of atomic equality relations. An example of E is $(C_1 = C_3 \wedge C_1 = C_4 \wedge C_2 = C_3)$. We say that M satisfies E at state q if there is some input $w = a_1^{i_1} \ldots a_n^{i_n}$ such that M on input w, enters some configuration where the state is q and the counter values satisfy the relation E. For convenience, when q is understood, we simply say M satisfies E. Also, when we say $(i_1, \ldots, i_n)$ is the input to M, we mean that the word $a_1^{i_1} \ldots a_n^{i_n}$ is the input.

Since M does not necessarily halt, a configuration that satisfies E can be an intermediate configuration of a possibly infinite computation. Note also that M can satisfy E many times during the computation. We are interested in the following *reachability problem*:

Given: A 2FAMC M and an equality relation E.
Question: Does M satisfy E ?

Two obvious generalizations of the above problem are: (1) when the input w to M does not come from a bounded language, and (2) when E is an arbitrary Presburger relation (note that an equality relation is a special form of Presburger relation).

A special case of a 2FAMC is one that always halts. Then the two-way input of such a machine will necessarily be finite-crossing. Therefore, one can use Theorem 2(d) and Theorem 4(a) to show that the reachability problem is decidable, even when E is any Presburger relation and the machine is not over a bounded language:

Theorem 5. *It is decidable to determine, given a halting 2FAMC (not necessarily over a bounded language) and any Presburger relation E, whether M satisfies E.*

However, when M does not always halt, the problem is open. More precisely:

Open Problem 1: Is the reachability problem decidable when M does not always halt and E is an equality relation? If the answer is yes, then what about when E is an arbitrary Presburger relation and/or when M is not over a bounded language?

4 2FAMC and Diophantine Equations

The open problem concerning 2FAMC is intimately connected to Diophantine equations. Consider a system S_1 consisting of the following $(k+m)$ equations:

$A_i = B_i,\ i = 1, \ldots, k$

$yF_i = G_i,\ i = 1, \ldots, m$

where A_i, B_i, F_i, G_i are linear polynomials in nonnegative integer variables $x_1, \ldots, x_n$ with integer coefficients. Hence these polynomials are of the form $a_0 + a_1x_1 + \ldots + a_nx_n$, where each a_i is an integer (positive, negative, or zero). We say that S_1 has a solution if there are nonnegative integers $y, x_1, \ldots, x_n$ satisfying S_1.

The solvability of the system above can be reduced to the solvability of a simpler class S_2 of equations, where we only have the second type of equations:

$yF_i = G_i,\ i = 1, \ldots, m$

and F_i, G_i are *positive* linear polynomials, i.e., the coefficients are now *nonnegative* integers (positive or zero). We shall refer to the systems above as type 1 and type 2.

Lemma 1. *S_1 is solvable if and only if S_2 is solvable.*

Proof. We only need to prove the "if" part. So let S_1 be a type 1 system. Without loss of generality, we may assume that A_i, B_i, F_i, G_i are nonnegative for all nonnegative integral values of $x_1, ..., x_n$. The idea is to use a "semilinear transformation". We only illustrate the transformation for the case when $k = m = 1$. For simplicity, call the polynomials A, B, F, G. The generalization for any k and m is straightforward.

Let $a, b, f, g, d_1, ..., d_n$ be distinct symbols and define the language $L = \{a^{v_A}b^{v_B}f^{v_F}g^{v_G}d_1^{i_1}...d_n^{i_n} \mid i_1, ..., i_n \in \mathbf{N}, v_A = A(i_1, ..., i_n), v_B = B(i_1, ..., i_n), v_F = F(i_1, ..., i_n), v_G = G(i_1, ..., i_n), v_A = v_B\}$. We can easily construct a 1-crossing 2NCM(k) for some k (i.e., a one-way nondeterministic machine M with k reversal-bounded counters) accepting L. From Theorem 3, $p(L(M))$ is a semilinear set Q effectively computable from M. Assume first that Q is a linear set. Then, clearly, v_F and v_G can be written as positive linear polynomials in some other nonnegative integer variables $x'_1 ..., x'_{n'}$ for some n', and this type 2 system has a solution if and only if S_1 has a solution. If Q is a union of linear sets, then we can construct a finite number of type 2 systems such that one of them has a solution if and only if S_1 has a solution. ∎

It is not known whether there is an algorithm to determine, given a system S (either type 1 or type 2), whether it has a solution. It turns out that this problem is equivalent to the reachability problem for 2FAMC.

Lemma 2. *Given a system S, we can effectively construct a 2FAMC M and an equality relation E such that S has a solution if and only if M satisfies E.*

Proof. By Lemma 1, we need only consider a type 2 system. The input to M is a tuple of nonnegative integers $(i_1, \ldots, i_n)$. M has monotonic counters C_{F_i}, C_{G_i} (initially 0 and to be defined below) and operates as follows:

Step 1: For $i = 1, \ldots, m$, M reads the two-way input and computes $G_i(i_1, \ldots, i_n)$ into counter C_{G_i}.

Step 2: For $i = 1, \ldots, m$, M reads the two-way input and computes $F_i(i_1, \ldots, i_n)$ while adding it to counter C_{F_i}. Goto *Step 2*.

Note that *Step 2* is an infinite loop. Let E to be the conjunction of the atomic equality relations $C_{F_i} = C_{G_i}$ $(i = 1, ..., m)$, and q be the state each time when M just enters *Step 2*. Clearly, S has a solution if and only if M satisfies E. ∎

We will show the converse of Lemma 2, but first we prove a positive result. A *semi-equality relation* E is a set of equality relations. Thus, $E = \{E_1, \ldots, E_m\}$, where each E_i is an equality relation (i.e., a conjunction of atomic equality relations). Note that when $m = 1$, E is simply an equality relation. E is of *width* k if each E_i is a conjunction of at most k atomic equality relations. Given M, a semi-equality relation $E = \{E_1, \ldots, E_m\}$, and states $q_1, \ldots, q_m$ (not necessarily distinct), M satisfies E if there is a tuple $(i_1, \ldots, i_n)$ such that M on input $(i_1, \ldots, i_n)$ has a computation where the counter values satisfy each E_i at state q_i during the computation (not necessarily at the same time, and not necessarily in the given order). To avoid writing so many subscripts, when $\{q_1, \ldots, q_m\}$ is understood, we simply say M satisfies E.

Theorem 6. *It is decidable to determine, given a 2FAMC M and a width 1 semi-equality relation E, whether M satisfies E.*

Proof. Let M be a machine with a two-way input $(i_1, \ldots, i_n)$ and k monotonic counters $C_1, \ldots, C_k$. Each E_i in $E = \{E_1, \ldots, E_m\}$ is an atomic equality relation. Since M is deterministic, if M crosses a cell twice in the same direction and in the same state, then it is in a loop, and M will repeat the loop forever. If M satisfies E_i at the specified state q_i, then M satisfies E_i either before entering the loop or during the loop (after the loop is executed for 0 or more times). First consider the case when M satisfies each E_i during the loop. We construct a nondeterministic machine M' that simulates M as follows. M' is equipped with $m+2$ sets of monotonic counters: $C_1^j, \ldots, C_k^j$, $1 \leq j \leq m+2$ that are initially 0. On an input $(i_1, \ldots, i_n)$ of M, M' works on the same input on which there is one position marked with $\diamond$ and m positions marked with $\diamond_1, \ldots, \diamond_m$, respectively. M' simulates M faithfully and ignores the marks. During this phase (called the first phase), M' increments the first set of counters only, according to the transition rules of M. At some moment (chosen nondeterministically and not necessarily the first time) when M' is reading the cell marked by $\diamond$, it remembers the current state and head direction. It continues the simulation, while incrementing the second set of counters instead of the first set, until M' returns to the same cell marked by $\diamond$ with the state and direction being as remembered. This is called the second phase. After this point, M' continues the simulation, while incrementing the last m sets of counters and keeping the first two sets unchanged. During this third phase, for each $1 \leq i \leq m$, at some moment (again, not necessarily the first time) when M' reaches the cell marked by $\diamond_i$, and specified state q_i, M' stops incrementing the $i+2$-th set of counters afterwards. M' *terminates* if the counters are stopped for each $1 \leq i \leq m$. Clearly, M' is restricted in such a way that it never crosses a cell for more than $2 \cdot |S| + 1$ times ($|S|$ is the number of states in M). Thus, the two-way input head is finite-crossing. We use $c_1^j, \ldots, c_k^j$, $1 \leq j \leq m+2$, to denote the counter values when M' terminates on input $(i_1, \ldots, i_n)$ with marks $\diamond, \diamond_1, \ldots, \diamond_m$. We now construct from M' another machine M''whose input is the input to M' padded at the end with $(c_1^1, \ldots, c_k^1, \ldots, c_1^{m+2}, \ldots, c_k^{m+2})$. M'' simulates M' and when M' terminates, M'' reverses the counters and accepts if their values correspond to the values padded on the input. Now M'' is a finite-crossing reversal-bounded counter machine. By Theorem 3 and the fact that semilinear sets are closed under projection (quantification), the set of tuples $(c_1^1, \ldots, c_k^1, \ldots, c_1^{m+2}, \ldots, c_k^{m+2})$ accepted by M'' is a semilinear set, Q, i.e., a union of linear sets. Assume first that Q is a linear set. Then each c_i^j can be written as a positive linear polynomial $f_i^j(x_1, \ldots, x_s)$. (Thus, the coefficients in f_i^j are nonnegative. Note also that the polynomial can simply be a nonnegative integer constant, even zero.) Now E_i is satisfied if for each $1 \leq i \leq m$, (assume E_i is $c_{i_1} = c_{i_2}$), there is a number $y_i \geq 0$ such that

$$c_{i_1}^1 + y_i \cdot c_{i_1}^2 + c_{i_1}^{i+2} = c_{i_2}^1 + y_i \cdot c_{i_2}^2 + c_{i_2}^{i+2}. \tag{1}$$

Substituting the positive linear polynomials into the c_i^j's and rearranging, we get a system S of equations of the form $y_i F_i = G_i$, $i = 1, \ldots, m$, which has a solution in $y_1, \ldots, y_m, x_1, \ldots, x_s$ if and only if M satisfies E. (Note that the F_i's and G_i's are linear polynomials, but may not necessarily be positive linear polynomials.)

To handle the general case when some of the E_i's are satisfied before M enters the loop, the construction of M' and M'' has to be modified so that, e.g., if E_i (assume it is $c_{i_1} = c_{i_2}$) is satisfied before the loop, then

instead of (1) above, we would have:

$$c^1_{i_1} = c^1_{i_2}. \tag{2}$$

Thus, for the general case, for some of the $i's$, we will have equation of the form $A_i = B_i$ instead of $y_iF_i = G_i$ in the system S.

If E is a finite union of linear sets $Q_1, Q_2, \ldots$, we obtain the corresponding systems $S_1, S_2, \ldots$ such that M satisfies E if and only if one of the S_i's has a solution. The result now follows from the following theorem in [7]. ∎

Theorem 7. *It is decidable to determine, given a system S of equations of the form $A_i = B_i$ or of the form $y_iF_i = G_i$ where A_i, B_i, F_i, G_i are linear polynomials in $x_1, \ldots, x_n$, whether S has a nonnegative integer solution in the y_i's and the x_i's.*

From the proofs of Lemma 2 and Theorem 6, we get:

Corollary 1. *Given a system S of equations of the form $A_i = B_i$ or of the form $y_iF_i = G_i$, we can effectively construct a 2FAMC M and a width 1 semi-equality relation E such that S has a solution if and only if M satisfies E. Conversely, given a 2FAMC M and a width 1 semi-equality relation E, we can effectively construct a finite number of systems S of equations such that M satisfies E if and only if one of the S's has a solution.*

The following is the converse of Lemma 2.

Lemma 3. *Given a 2FAMC and an equality relation E, we can effectively construct a finite number of systems S of equations of the form $A_i = B_i$ or of the form $yF_i = G_i$ such that M satisfies E if and only if one of the S's has a solution in the nonnegative integers $y, x_1, \ldots, x_n$.*

Proof. Since E is a conjunction of atomic equality relations $E_1, \ldots, E_m$ that must be satisfied at the same time in a specified state q, $q_1 = \ldots = q_m = q$, and $\diamond_1, \ldots, \diamond_m$ must appear in the same position. Hence, there is only one state and only one marked position. Thus, in the the proof of Theorem 6, we should have: $yF_i = G_i$, $i = 1, \ldots, m$. (I.e., the same variable y appears in each equation.) ∎

The following, which is equivalent to Open Problem 1, follows from Lemmas 2 and 3.

Open Problem 1': Is it decidable to determine, given a system S consisting of equations of the form $A_i = B_i$ or of the form $yF_i = G_i$, whether S has a solution in the nonnegative integers, $y, x_1, \ldots, x_n$?

Remark: It is easy to verify that Theorem 6 and Lemma 3 still hold even when the input w to M does not come from a bounded language.

Turning now to the case when the semi-equality relation is width 2, we have the following rather surprising result.

Theorem 8. *It is undecidable to determine, given a 2FAMC M and a width 2 semi-equality relation E, whether M satisfies E. The result holds even when each counter is involved in only one atomic comparison in E.*

Proof. The proof is an intricate reduction to Hilbert's Tenth Problem. Given a Diophantine polynomial p, we construct a deterministic machine M with a two-way input (with endmarkers) over a bounded language and a finite number of monotonic counters and a width 2 semi-equality relation E such that M satisfies E if and only if p has a nonnegative integral solution.

Let $p = (t_1 + \ldots + t_k) - (t_{k+1} + \ldots + t_q) + d$ where each term t_i is of the form $c\alpha$, where c is a positive integer coefficient, α is a product of nonnegative integer variables, and d is an integer (positive, negative, or zero). We may assume that $c = 1$ (otherwise, we can just write the term α for c times).

The input alphabet of M consists of many symbols. For each term $t = X_1 \ldots X_m$ in the polynomial, where each X_j is a variable, its "factors" are:

$$X_1, \ldots, X_m, X_1X_2, X_1X_2X_3, \ldots, X_1X_2X_3 \ldots X_m$$

(deleting duplications). For example, if the term is x^2yz, then its factors are: x, y, z, x^2, x^2y, x^2yz. We associate a distinct symbol for every factor in the term. So, e.g., for the term x^2yz, we require distinct *original input symbols* (using square brackets to denote a symbol):

$$[x], [y], [z], [x^2], [x^2y], [x^2yz].$$

Since there are a finite number of terms in p, we thus create a finite number of original input symbols. For each original symbol a thus defined, we create finitely many new input symbols $a', a'', \ldots$. The number of these new symbols wrt a is at most the number of common factors a represents in the terms in p. The purpose of such symbols will become clear later. Thus, e.g., we will also have symbols $[x]', [x]'', \ldots, [x^2]', [x^2]'', \ldots$. We refer to these symbols as *primed* input symbols. We then associate with every input symbol b (an original or primed input symbol), a monotonic counter $C(b)$; e.g., $C([x^2yz])$, $C([x^2yz]')$, $\ldots$ are counters. For every monotonic counter C thus defined, we also define another counter D, e.g., $D([x^2yz])$. We shall refer to these counters as D-counters. There are also two special counters, C_{p^+} and C_{p^-}. For each input symbol b, we denote by $V(b)$ any string of the form b^j for some j (i.e., a string of j b's). We refer to $V(b)$ as the value of b.

If $b_1, \ldots, b_r$ are all the input symbols (in some lexicographic order), the input to M is a string of the form $V(b_1) \ldots V(b_r)$.

We now describe the operation of M. We also include the elements in the semi-equality relation E to be satisfied.

Phase 1:

In this phase, M first scans the input and stores $V(b_i)$ to counter $C(b_i)$ for $i = 1, \ldots, r$. Then M computes $V(t_1) + \ldots + V(t_k)$ in counter C_{p^+} and $V(t_{k+1}) + \ldots + V(t_q)$ in counter C_{p^-}. It also adds $|d|$ to the first (resp., second) counter if d is nonnegative (resp., negative). Note that all D-counters are zero at this point.

Relation: $C_{p^+} = C_{p^-}$, $C(a_j) = C(a_j') = C(a_j'') \ldots$ for each original input symbol a_j.

This relation (which is an element in E) checks that $V(a_j) = V(a_j') = V(a_j'') = \ldots$ for each original input symbol a_j. Also, assuming that the terms are verified (to be described below), the polynomial has a nonnegative integer solution.

Since we want to make sure that no counter is involved in more than one comparison, we can add additional counters that are used only in this phase: $X_j', X_j'', \ldots$ and $Y_j', Y_j'', \ldots$ for each j. Then at the beginning, M also stores $V(a_j)$ to $X_j', X_j'', \ldots$ and $V(a_j')$ to Y_i', $V(a_j'')$ to Y_j'', ... The relation $C(a_j) = C(a_j') = C(a_j'') \ldots$ would then be replaced by:

$$X_j' = Y_j', X_j'' = Y_j'', \ldots$$

Note that the above atomic comparisons need not all be satisfied at the same time (so long as each is satisfied at some time during the computation). We will see that $C(a_j), C(a_j'), C(a_j''), \ldots$ are involved in comparisons in Phase 2, so we don't want to use them for comparisons in this phase.

Phase 2:

In this phase, M verifies that the value of each original input symbol is correct. This phase does *not* halt. It iterates a "process" that we describe below.

We first describe the operation of M on one original input symbol. Suppose the symbol is $[x^2yz]$. M has to verify for x^2, x^2y, and x^2yz. That is, we only need to verify the following constraints:

$V([x^2]) = V([x]) \cdot V([x])$,
$V([x^2y]) = V([x^2]) \cdot V([y])$,
$V([x^2yz]) = V([x^2y]) \cdot V([z])$.
Once this is done, we can safely say that $V([x^2yz]) = V([x])^2 \cdot V([y]) \cdot V([z])$; i.e., the input value of the original input symbol $[x^2yz]$ correctly corresponds to the values for $[x], [y], [z]$. Note that $V([x]), V([y]), V([z])$, $V([x^2]), V([x^2y]), V([x^2yz])$ have been stored in their associated counters (in Phase 1). Also, we may assume, from Phase 1, that there are "copies" of these values in other (primed) counters. M will do the verification simultaneously by making an infinite left-to-right sweeps of the input. We describe the verification of $V([x^2])$ first.

In each left-to-right sweep of the input, M increments counter $D([x^2])$ by value $V([x])$ (which is represented on the input) and counter $D([x])$ by 1. Note that eventually, the value of $D([x])$ will reach $V([x])$ (stored in $C([x])$ in Phase 1) and at that time the value of $D([x^2])$ will be $V([x])^2$ (stored in $C([x^2])$ in Phase 1).

Relation: $C([x]) = D([x]) \ \wedge \ C([x^2]) = D([x^2])$.

Note that the above relation is satisfied if and only if $V([x^2]) = (V[x])^2$. Once this has been verified, the values of these counters $D([x])$ and $D([x^2])$ at other times (which will keep on increasing) are no longer relevant.

Simultaneously, during each left-to-right sweep:

1. M increments counter $D([x^2y]')$ by value $V([y])$ and counter $D([x^2]')$ by 1.

Relation: $C([x^2]') = D([x^2]') \ \wedge \ C([x^2y]') = D([x^2y]')$.

This relation verifies that $V([x^2y]) = V([x^2]) \cdot V([y])$. Clearly, $C([x^2]') = C([x^2])$ and $C([x^2y]') = C([x^2y])$, since the values of these counters that were loaded in Phase 1 do not change during the computation. Actually, we could have used $C([x^2])$ and $C([x^2y])$ instead in the above relation, but to avoid confusion we use the "copies". However, since the counter $D([x^2])$ is no longer zero and will increase during the computation, we *need* to use a new counter (initially zero) $D([x^2]')$. Though we use $D([x^2y]')$ in the relation, we could have used $D([x^2y])$ also. Note that using new counters (i.e., the primed) has the added advantage in that in the relations, a counter is involved in at most one comparison.

2. M increments counter $D([x^2yz]'')$ by value $V([z])$ and counter $D([x^2y]'')$ by 1.

Relation: $C([x^2y]'') = D([x^2y]'') \ \wedge \ C([x^2yz]'') = D([x^2yz]'')$.

This relation verifies that $V([x^2yz]) = V([x^2y]) \cdot V([z])$. After this point, we are convinced that $V([x^2yz])$ encodes a desired value for $[x^2yz]$.

The ideas above easily generalize to the verification of several original input symbols simultaneously by making left-to-right sweeps of the input. Since the terms in p will have common factors, we need multiple copies of the inputs (this is the purpose for the primed inputs) and primed C counters and D counters. The semi-equality relation E is the set of all equality relations defined above. Let q be the state M enters at the end of every left-to-right sweep in Phase 2. Then M satisfies each width 1 equality relation (defined in Phase 1) and each width 2 equality relation (defined in Phase 2) when it is in state q (but perhaps at different sweeps) if and only if the polynomial p has a nonnegative integer solution. The result follows from the undecidability of Hilbert's Tenth Problem. ∎

Now consider the following system of equations, S, which we call a width 2 system:

$A_i = B_i, \ \ i = 1, \ldots, k$
$y_i F_i = G_i \ \wedge \ y_i H_i = I_i, \ \ i = 1, \ldots, m$
where $A_i, B_i, F_i, G_i, H_i, I_i$ are linear polynomials in nonnegative variables $x_1, \ldots, x_n$. Note that each y_i is involved in exactly two equations. We say that S has a solution if there are nonnegative integers $y_1, \ldots, y_m, x_1, \ldots, x_n$ satisfying S.

Corollary 2. *It is undecidable to determine, given a width 2 system S, whether it has a solution.*

Proof. A close look at the proof of Theorem 8 shows that the machine M that we constructed operates on an input (from a bounded language) in a regular pattern. Let the input be parameterized by nonnegative integer variables $x_1, \ldots, x_n$. Call all the counters introduced in the proof $C_1, \ldots, C_t$. Then we see that the value of each counter C_i at the end of iteration y is $yF_i(x_1, \ldots, x_n)$ for some linear polynomial F_i. The set of relations in E that must be satisfied (in state q) consists of width 1 atomic equality relations defined in Phase 1, and width 2 equality relations of the form $(C_i = C_j \ \wedge \ C_r = C_s)$ defined Phase 2. The result follows from Theorem 8. ∎

Corollary 3. *Given a width 2 system S, we can effectively construct a 2FAMC M and a width 2 semi-equality relation E such that S has a solution if and only if M satisfies E. Conversely, given a 2FAMC M and a width 2 semi-equality relation E, we can effectively construct a finite number of width 2 systems S such that M satisfies E if and only if one of the S's has a solution.*

Proof. The first part follows from the proof of Corollary 2. The second part follows from the proof of Theorem 6 (see also the proof of Lemma 3). ∎

5 Nondeterministic 2FAMC

In this section, we study the reachability problem when the 2FAMC is nondeterministic. We begin with the following theorem.

Theorem 9. *1. There is a fixed k such that it is undecidable to determine, given a nondeterministic 2FAMC M with k monotonic counters and an equality relation E, whether M satisfies E. The result also holds for the case when M always halts.*
2. It is decidable to determine, given a nondeterministic 2FAMC M over a unary input alphabet (but no restriction on the number of monotonic counters) and a Presburger relation E, whether M satisfies E.

Proof. We first prove Part 1. From Theorem 1, there is a fixed r such the emptiness problem for 2DCM(2,r) over bounded languages is undecidable. Let M be such an automaton. Clearly, we can convert M to an equivalent automaton M' which has $2(1 + \frac{r-1}{2})$ 1-reversal counters where each counter starts at zero, and on input w, M' accepts if and only if it halts with all counters zero. To insure this, we can add to M' a dummy counter which is incremented at the beginning and only decremented, i.e., becomes zero when the input is accepted. Suppose M' has k 1-reversal counters, $C_1, ..., C_k$ (one of these is the dummy counter). Note that k is fixed since r is fixed. We modify M' to a nondeterministic 2FAMC M'' with $2k$ monotonic counters: $C_1^+, C_1^-, ..., C_k^+, C_k^-$, where C_i^+ and C_i^- are associated with counter C_i. M'' on input w simulates the computation of M' , first using counter C_i^+ to simulate C_i when the latter is in a nondecreasing mode. When counter C_i reverses (thus entering the nonincreasing mode), M'' continues the simulation but using counter C_i^-: incrementing this counter when M' decrements C_i. At some time during the computation (which may be different for each i), M' guesses that C_i has reached the zero value. From that point on, M'' will no longer use counters C_i^+ and C_i^-, but continues the simulation. When M'' has guessed that $C_i^+ = C_i^-$ for all i's (note that for sure the two counters corresponding to the dummy counter are equal), it halts in a unique state f. Clearly, w is accepted by M' if and only if M'' on w can reach a configuration with $C_i^+ = C_i^-$ for $i = 1, ..., k$ (this is relation E) in state f.

For Part 2, given an 2FAMC M with monotonic counters $C_1, ..., C_k$, we construct a 2NCM(1) (i.e., two-way nondeterministic finite automaton with reversal-bounded counters) over a unary input M'. M' simulates M faithfully. At some point, M' guesses that the values of the monotonic counters satisfy the relation E. M' then uses another set of counters to verify that this is the case, and accepts. The result follows from the decidability of the emptiness problem for 2NCM(1) over unary input (Theorem 2(c)) and and the fact that a Presburger relation on $C_1, ..., C_k$ can be verified using additional reversal-bounded counters (Theorem 4(a)). ∎

In Theorem 9, Part 2, the relation E is defined over monotonic counters $C_1, \ldots, C_k$ and a specified state. In fact, the theorem still holds when the relation is defined over $C_1, \ldots, C_k, i_1, \ldots, i_n$, the input head position, and the state. This is because $n+2$ additional monotonic counters can be added to "store" the values of $i_1, \ldots, i_n$, the input head position, and the state at the time when the test for E is performed. The next result looks at the case when $k = 1$.

Theorem 10. *The reachability problem is decidable when M is a nondeterministic 2FAMC and $k = 1$, i.e., there is only one monotonic counter. The result holds even when the relation E is a Presburger relation that involves the state, the input head position, the input $(i_1, \ldots, i_n)$, and the monotonic counter.*

Proof. Given M, we construct a 2NCM(1) (i.e., a two-way nondeterministic finite automaton with one reversal-bounded counter) M' which accepts a nonempty language if and only if M satisfies the relation E. Let $\Sigma = \{a_1, \ldots, a_n\}$ be the input alphabet of M. The input alphabet of M' is $\Delta = \{a_1, \ldots, a_n, \diamond, b, c, d\}$, where $\diamond, b, c, d$ are new symbols. Given input y, M' first checks that y has exactly one occurrence of $\diamond$, and y with this $\diamond$ deleted is a string of the form $a_1^{i_1} \ldots a_n^{i_n} b^j c^k d^m$. M' then simulates M faithfully on $a_1^{i_1} \ldots a_n^{i_n}$, using its reversal-bounded counter to simulate the monotonic counter of M. During the simulation, when M' sees $\diamond$, it nondeterministically either ignores it and continues the simulation, or enters the testing phase. When M' decides to enter the testing phase (thus its input head is on $\diamond$), it checks the following: (1) the number j of b's is the input head position (i.e., the distance of $\diamond$ from the left end of the input), (2) the number k of c's is the value of the counter, and (3) the number m of d's represents the current state. M' accepts the input y if $i_1, \ldots, i_n, j, k, m$ satisfy the Presburger relation E. Note that M' can check the Presburger relation by Theorem 4(b). Clearly, M' accepts a bounded language. The result follows since the emptiness problem for 2NCM(1)'s over bounded languages is decidable (Theorem 2(b)). ∎

It is open whether Theorem 10 holds when $k = 2$. But consider a nondeterministic machine M with k monotonic counters, $C_1, \ldots, C_k$, *ordered* in that for $2 \leq i \leq k$, when C_i is first incremented, $C_1, \ldots, C_{i-1}$ can no longer change in value. The following corollary generalizes Theorem 10:

Corollary 4. *It is decidable to determine, given a nondeterministic machine M with k monotonic ordered counters and an arbitrary Presburger relation E (over the state, the input head position, $i_1, \ldots, i_n$, and the k counters), whether M satisfies E.*

Proof. Since the k counters are ordered, it is straightforward to generalize the construction in the proof of Theorem 10. M' will now have alphabet $\Delta = \{a_1, \ldots, a_n, \diamond_1, \ldots, \diamond_k, b, c_1, \ldots, c_k, d\}$. The marker $\diamond_i$ is used to remember the input head position when M goes from counter C_i to C_{i+1} $(1 \leq i < k)$. As before, marker $\diamond_k$ is used to record the position of the input head prior to testing E. Symbol c_i is used to record the value of counter C_i. We leave the details to the reader. ∎

Remark: It is easy to check that Theorem 10, and Corollary 4 remain valid when the monotonic counter is replaced by a reversal-bounded counter.

Open Problem 3: In Theorem 9, Part 1, it would be interesting to find the smallest k for which the problem is undecidable (even the case $k = 2$ is open).

When E is a semi-equality relation, we have the following result which contrasts Theorem 6:

Theorem 11. *It is undecidable to determine, given a nondeterministic 2FAMC M and a width 1 semi-equality relation E, whether M satisfies E.*

Proof. M operates in two phases. The first phase implements exactly the first phase in the proof of Theorem 8. From the discussion in that proof, it is clear that the semi-equality relation to be satisfied with respect to this phase has width 1.

The second phase of M implements the second phase in the proof of Theorem 8, i.e., it verifies the terms $t_1, \ldots, t_q$, but the verification is done one at a time: first t_1, then t_2, etc. (instead of simultaneously as is done in Theorem 8). So, e.g. if $t_1 = x^2yz$, then the following code (using the notation in the proof of Theorem 8) verifies this term. The label 4 is the start of the code for the term t_2.

$1:$ $D([x^2]) := D([x^2]) + V([x])$;
$D([x]) := D([x]) + 1$;
goto 1 or 2;
$2:$ $D([x^2y]') := D([x^2y]') + V([y])$;
$D([x^2]') := D([x^2]') + 1$;
goto 2 or 3;
$3:$ $D([x^2yz]'') := D([x^2yz]'') + V([z])$;
$D([x^2y]'') := D([x^2y]'') + 1$;
goto 3 or 4;

$\vdots$

Again, the semi-equality relation to be checked with respect to this second phase has width 1. For example, with respect to term t_1, the atomic equality relations are: $C([x]) = D([x]), C([x^2]') = D([x^2]')$, and $C([x^2y]'') = D([x^2y]'')$.

Let q be state of M at the end of the computation of all the terms. Then the semi-equality relation $E = \{E_1, \ldots, E_m\}$ to be satisfied by M at state q has width 1, i.e., each E_i is an atomic equality relation. ∎

Next, consider the following problem, where M is a 2NCM(1) over $a_1^* \ldots a_n^*$ and $E(x_1, \ldots, x_m)$ is a Presburger formula. We use $C[t]$ to denote the value of counter C at time t ("time" is used to count the total number of moves). Let $q_1, \ldots, q_m$ be given states. We say that M *m-satisfies* E if for some input $(i_1, \ldots, i_n)$, there is a computation of M such that for some $t_1 < \ldots < t_m$, $E(C[t_1], \ldots, C[t_m])$ is true and for each q_i $(1 \leq i \leq m)$, M is at q_i when the current time is t_i. We show that this problem is decidable. Notice that this decidability does not simply follow from Corollary 4.

Theorem 12. *It is decidable to determine, given a 2NCM(1) M and a Presburger formula $E(x_1, \ldots, x_m)$, whether M m-satisfies E.*

Proof. We construct from M and E another 2NCM(1) M' over input alphabet $\{a_1, \ldots, a_n, b_1, \ldots, b_m, d_1, \ldots, d_m\}$. An input w to M' is valid if: (a) w has exactly one occurrence of d_k for each $k = 1, \ldots, m$; (b) w with the d_k's deleted results in a string of the form $a_1^{i_1} \ldots a_n^{i_n} b_1^{j_1} \ldots b_m^{j_m}$; (c) All the d_k's occur before the occurrence of any b_i in w. We refer to $d_1, \ldots, d_k$ as "markers".

M' first checks that input w is valid. Then it simulates the computation of M on $(i_1, \ldots, i_n)$ ignoring the markers. At some time t_1 during the computation, chosen nondeterministically, M' will be in some position within i_r. M' checks that the symbol directly to the right of this position is marker d_1 (and the state of M is q_1). (Note that M' may have seen this marker many times earlier but ignored it during the simulation until it decided that it has reached time t_1.) This marker is needed so that M' can return to this position after doing the following: M' moves its input head to the right and checks that j_1 is equal to the value of the counter C (if not, it halts and rejects). If it checks okay, then M' restores the value of the counter and moves its input head to marker d_1. Then M' resumes the simulation of M, and in the same way nondeterministically guesses

times $t_2, \ldots, t_m$ and checks that the values of counter C at these times are $j_2, \ldots, j_m$ (as well as the states are $q_2, \ldots, q_m$), respectively. Finally, M' verifies that $E(j_1, \ldots, j_m)$ is true.

Clearly, M m-satisfies $E(C[t_1], \ldots, C[t_m])$ for some $i_1, \ldots, i_n, t_1, \ldots, t_m$ if and only if M' accepts some input w. The result now follows since emptiness for 2NCM(1) over bounded languages is decidable by Theorem 2(b). ∎

When the times $t_1, \ldots, t_m$ are involved in the Presburger formula, i.e., we now have $E(x_1, \ldots, x_m, t_1, \ldots, t_m)$, then the above problem is undecidable.

Theorem 13. *It is undecidable to determine, given a 2NCM(1) M and a Presburger formula $E(x_1, \ldots, x_m, t_1, \ldots, t_m)$, whether M m-satisfies E.*

Proof. It is known that, in general, a system of quadratic Diophantine equations is unsolvable [8]. Hence, it is undecidable to determine, given an n and a Presburger formula

$$P(A_1, \ldots, A_n, A_1A_1, \ldots, A_1A_n, \ldots, A_iA_i, \ldots, A_iA_n, \ldots, A_{n-1}A_n, A_nA_n) \tag{3}$$

(note that we write P in terms of the A_i's and the A_iA_j's, $1 \leq i \leq j \leq n$), whether it has a nonnegative integer solution in $A_1, \ldots, A_n$. Now, we construct an m, a 2NCM(1) M and an E such that (3) has a solution if and only if M m-satisfies E. The result then follows.

M operates on input $(A_1, \ldots, A_n)$ in two phases as follows with the counter initially being 0. In the first phase, for each $i = 1, \ldots, n$, M increments the counter to A_i while reading the block of A_i (and enters the state q_i – we use t_i to denote the current time) and then decrements the counter to 0. As a result, $C[t_i] = A_i$ $(1 \leq i \leq n)$. Suppose that the state is now q_{n+1} and the current time is t_{n+1}. In the second phase, from $i = 1$ to n, M executes the following subroutine:

1. $j := i$;
2. Repeat below for 0 or more times (nondeterministically chosen):
   2.1. Read the segment of A_i on the input from left to the right and back;
   2.2. Increment the counter by 1;
3. Decrement the counter to 0;
4. $j := j + 1$;
5. If $j > n$ then exit this subroutine else goto 2.

Let q_{ij} be the state of M when it is about to execute step 3. Note that each q_{ij} is visited only once during M's computation. We use T_{ij} to denote the time when M visits q_{ij} (for simplicity and without loss of generality, we only count the times spent in step 2 and step 3). Assume that $\wedge_{i \leq j} C[T_{ij}] = A_j$. Under this assumption, the loop in step 2 must be repeated for A_j times for each i and j. For $1 \leq i \leq n$ and $i \leq j < n$, $T_{i(j+1)} - T_{ij} = A_j + (2A_i + 1)A_{j+1}$. This is because, between time T_{ij} and time $T_{i(j+1)}$, M executes step 3 (takes A_j time units) and executes the loop in step 2 for A_{j+1} times (each loop takes $2A_i$ time units in step 2.1 and one time unit in step 2.2). Similarly, for $1 \leq i < n$, $T_{(i+1)(i+1)} - T_{in} = A_n + (2A_{i+1} + 1)A_{i+1}$, and $T_{11} = (2A_1 + 1)A_1 + t_{n+1}$ (t_{n+1} is the time when the second phase started). Therefore, each $A_iA_j, i \leq j$, can be expressed as a linear combination (with positive, negative, zero coefficients) of the T_{ij}'s and A_i's. Hence, combining the result of phase 1, each A_iA_j as well as each A_i can be expressed as a linear combination of the T_{ij}'s, $C[t_i]$'s, and t_{n+1}. Substituting each A_iA_j and each A_i in the conjunction of the assumption and (3) with the linear combinations representing them, it is not hard to obtain an E with $m = (n + 1) + \frac{n(n+1)}{2}$ as required. Note that M is $m - 1$ reversal bounded. ∎

6 Separation Results for 2DCM(1) and 2NCM(1)

In this section we give a technique that uses decidability and undecidability results to show "separation" between language classes.

Clearly, the language $L = \{a^i b^j c^k d^m \mid i, j, k, m \geq 1, i \text{ divides } j, k \text{ divides } m\}$ can be accepted by a 2DCM(1) (deterministic two-way finite automaton with one reversal-bounded counter). The following theorem exhibits a simple language L that cannot be accepted by a 2DCM(1) or by a 2NCM(1). That L cannot be accepted might be obvious, intuitively; but a formal proof is not straightforward. The proof, provided below, is interesting as it does not use the usual pumping lemma, crossing sequence, or other combinatorial arguments.

Theorem 14. *The language $L = \{a^i b^j c^k d^m \mid i, j, k, m \geq 1, i \text{ divides } j, k \text{ divides } m, j/i = m/k\}$ cannot be accepted by a 2DCM(1). In fact, it cannot be accepted by a 2NCM(1).*

Proof. Assume that L can be accepted by a 2DCM(1) M_L. From Theorem 8, it is undecidable to determine, given a 2FACM M and a width 2 semi-equality relation E, whether M satisfies E.

Referring to the proof of Theorem 6, let L_Q be the bounded language consisting of tuples $(c_1^1, \ldots, c_k^1, \ldots, c_1^{m+2}, \ldots, c_k^{m+2})$ corresponding to the semilinear set Q. From Theorem 4(b), L_Q can be accepted by a 2DCM(1) M_{L_Q}. Now equation (1) in the proof of Theorem 6 is the following:

$c_{i_1}^1 + y_i \cdot c_{i_1}^2 + c_{i_1}^{i+2} = c_{i_2}^1 + y_i \cdot c_{i_2}^2 + c_{i_2}^{i+2}$

We can modify M_{L_Q} so that it also checks the above equation, i.e., checks that $c_{i_2}^1 + c_{i_2}^{i+2} - c_{i_1}^1 - c_{i_1}^{i+2}$ is divisible by $(c_{i_1}^2 - c_{i_2}^2)$ (assuming that both are positive). This can be done easily provided $(c_{i_1}^2 - c_{i_2}^2)$ is available on the input. But this value can be "padded" in L_Q, i.e., we modify the language L_Q into a language L'_Q, which is L_Q padded with differences of the form $(c_i^2 - c_j^2)$. But since E is width 2, each equality relation in E is a conjunction of two atomic equality relations. Hence, equation (1) will now consist of two equations with the "same" y_i on the left side of each equation. Since, by assumption, there is a 2DCM(1) M_L accepting the language L, we can incorporate the operation of M_L and further modify M_{L_Q} to check that the two equations are satisfied by the same y_i.

Hence, we can construct a 2DCM(1) that accepts a nonempty language if and only if M satisfies the width 2 semi-equality relation E. But this leads to a contradiction, since the emptiness problem for DCM(1)'s is decidable by Theorem 2(a). We note that L cannot also be accepted by a 2NCM(1), since the emptiness problem for these machines over bounded languages is also decidable by Theorem 2(b). ∎

Finally, we show that there is a language accepted by a 2NCM(1) that cannot be accepted by a 2DCM(1). Interestingly, one can prove this result by a reduction to the halting problem for Turing machines, as we show below. Consider only single-tape TMs Z over the alphabet $\{s_1, s_2, s_3\}$ (one symbol represents blank). We assume that these symbols are different from 0 and 1. Let $q_1, q_2, \ldots$ be the states, where q_1 is the initial state, and q_2 is the unique halting state.

Let r be a transition rule of the form $(q_i, a) \rightarrow (q_j, b, d)$, where d = 0 (1) represents left (right) move. We encode this rule by the string $E(r) = 1^i * a * 1^j * b * d$. If R is a set of rules = $\{r_1, \ldots, r_k\}$, let $E(R) = E(r_1)\%E(r_2)\%\ldots\%E(r_k)$. Note that R need not necessarily constitute a deterministic set of rules.

We represent a configuration of the TM on the tape as a string $w = u1^i v$, where u and v are strings in $\{s_1, s_2, s_3\}^*$. This represents the configuration where the tape content is uv, the read/write head is on the first symbol of v, and the state is q_i.

Let Σ be the alphabet $\{s_1, s_2, s_3, 0, 1, *, \%, \#\}$. Define the following language L_h over Σ as follows: A string $x\#w_1\#x\#w_2\ldots\#x\#w_k$ is in L_h if:

1. $x = E(R)$ is an encoding of a set of rules of a TM.
2. $w_1, w_2, \ldots, w_k$ is a halting sequence of configurations of the TM represented by $E(R)$; i.e., for each i, w_{i+1} is a configuration that results from configuration w_i using a rule in x.

Lemma 4. *Let L'_h be the complement of L_h, i.e., $L'_h = \Sigma^* - L_h$. We can effectively construct a one-way nondeterministic finite automaton with one reversal-bounded counter (1NCM(1)) M'_h accepting L'_h.*

Proof. M'_h, when given an input, nondeterministically guesses and executes one of the following:

1. M'_h checks and accepts if the input is not of the form $x_1\#w_1\#x_2\#w_2 \ldots \#x_k\#w_k$, where each x_i is an encoding of some set of TM rules and w_i is a configuration. M'_h does not need to use the counter here.
2. M'_h checks that some x_i is different from some x_j, where i and j are chosen nondeterministically. Here, M'_h makes one reversal on the counter.
3. M'_h nondeterministically chooses an i and checks that $w_i\#x_i\#w_{i+1}$ is not valid, i.e., w_{i+1} is not a valid successor of w_i according to the rules in x_i. To do this, M'_h remembers the symbol currently under the read/write head and its neighbor symbols and increments the counter to store the state in w_i (note that the state is encoded in unary), and then decrements the counter (which stored the state) to nondeterministically find an applicable rule in x_i. From x_i, M'_h remembers the symbol that is to replace the symbol under the read/write head, the direction of the move, and uses the counter again to store the "next" state. This way, M'_h can check if w_{i+1} is not a valid successor of w_i. Note that M'_h makes no more than 3 reversals on the counter.

It is straightforward to verify that M'_h accepts L'_h. ∎

Theorem 15. *There is a language accepted by a 1NCM(1) (and, hence, by a 2NCM(1)) that cannot be accepted by 2DCM(1).*

Proof. We show that L'_h cannot be accepted by a 2DCM(1). Suppose L'_h can be accepted by a 2DCM(1). Then since 2DCM(1) is closed under complementation [6], there exists a 2DCM(1) M_h accepting $L_h = \Sigma^* - L'_h$. We show that there exists an algorithm to decide the halting problem for TMs on blank tape, which is a contradiction. The algorithm works as follows:

1. Given a TM Z, let $E(R)$ be an encoding of its transition rules, R. (Note that there are several equivalent encodings, depending on how we order the rules. Choose one.)
2. Let F_Z be the language $E(R)\#\Sigma^*$. Clearly, F_Z is a regular set, and we can effectively construct a finite automaton A_Z accepting F_Z.
3. Construct from the finite automaton A_Z and the 2DCM(1) M_h (accepting L_h) a 2DCM M_Z which accepts $F_Z \cap L_h$.
4. Test if the language accepted by M_Z is empty.

It is clear that Z does not halt on blank tape if and only if the language accepted by M_Z is empty. The result now follows from the undecidability of the halting problem for TMs on blank tape and the fact that the emptiness problem for 2DCM(1)'s is decidable (Theorem 2 (a)). ∎

Remark: The "halting sequence of configurations of a TM" has been used before to investigate the undecidability of certain questions concerning language acceptors. What is new in the construction in the proof of Lemma 4 is that we use a single language (sort of universal) L_h to encode the halting sequences of configurations of all TMs. We needed a single language since, otherwise, the reduction to the halting problem in Theorem 15 will not work.

7 Conclusion

We introduced the model of a two-way finite automaton augmented with monotonic counters operating on inputs over a bounded language, and studied the decidability of the reachability problem for both the deterministic and nondeterministic varieties. In particular, for the deterministic case (and its variations), we showed the connection of the reachability problem to the solvability of some classes of quadratic Diophantine equations. Finally, we presented a new technique for separating language classes using decidability and undecidability results.

References

1. Z. Dang, O. Ibarra, and Z. Sun. On the emptiness problems for two-way nfa with one reversal-bounded counter. In *Proc. Thirteenth Int. Symp. on Algorithms and Computation*, volume 2518 of *Lecture Notes in Computer Science*, pages 103–114. Springer-Verlag, 2002.
2. S. Ginsburg and E. Spanier. Semigroups, presburger formulas, and languages. *Pacific J. of Mathematics*, 16:285–296, 1966.
3. E. M. Gurari and O. H. Ibarra. The complexity of decision problems for finite-turn multicounter machines. *Journal of Computer and System Sciences*, 22:220–229, 1981.
4. E. M. Gurari and O. H. Ibarra. Two-way counter machines and diophantine equations. *Journal of the ACM*, 29(3):863–873, 1982.
5. O. H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the ACM*, 25(1):116–133, January 1978.
6. O. H. Ibarra, T. Jiang, N. Tran, and H. Wang. New decidability results concerning two-way counter machines. *SIAM J. Comput.*, 24:123–137, 1995.
7. L. Lipshitz. The diophantine problem for addition and divisibility. *Transactions of AMS*, 235:271–283, 1978.
8. Y. V. Matiyasevich. *Hilbert's Tenth Problem*. MIT Press, 1993.
9. M. Minsky. Recursive unsolvability of Post's problem of Tag and other topics in the theory of Turing machines. *Ann. of Math.*, 74:437–455, 1961.