Real-Counter Automata and their Decision Problems * (Extended Abstract)

Zhe Dang[1] **, Oscar H. Ibarra[2], Pierluigi San Pietro[3], and Gaoyan Xie[1]

[1] School of Electrical Engineering and Computer Science
Washington State University, Pullman, WA 99164, USA
[2] Department of Computer Science, University of California, Santa Barbara, CA 93106, USA
[3] Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italia

Abstract. We introduce real-counter automata, which are two-way finite automata augmented with counters that take real values. In contrast to traditional word automata that accept sequences of symbols, real-counter automata accept real words that are bounded and closed real intervals delimited by a finite number of markers. We study the membership and emptiness problems for one-way/two-way real-counter automata as well as those automata further augmented with other unbounded storage devices such as integer-counters and pushdown stacks.

1 Introduction

An automaton is a finite-state language acceptor, that is possibly augmented with other unbounded storage devices like counters, stacks, and queues. Decision problems like membership and emptiness have been extensively studied in automata theory in the past 50 years. The membership problem is to decide whether a given word is accepted by an automaton, while the emptiness problem is to decide whether the language accepted by an automaton is empty. Studies on the decision problems have been one of the focuses in automata theory and have already benefited almost every area in computer science, including model-checking [7, 26] that seeks (semi-) automatic procedures to check whether a system design satisfies its requirements. Algorithmic solutions to decision problems like emptiness for various classes of automata (e.g., finite automata, Buchi automata, tree automata, pushdown automata, etc.) have become part of the theoretical foundation of model-checking finite-state/infinite-state systems. For instance, it is known that various model-checking problems such as LTL model-checking over finite-state transition systems and reachability for some infinite-state transition systems can be reduced to various emptiness problems (e.g., [26, 10]). Still, practitioners in formal specification/verification keep challenging automata theorists with new models emerging from verification applications. Some of the models, however, have not been well-studied in traditional automata theory. A typical example concerns the theory and fundamental verification techniques for analyzing hybrid transition systems containing

* The research of Zhe Dang and Gaoyan Xie was supported in part by NSF Grant CCF-0430531. The research of Oscar H. Ibarra has been supported in part by by NSF Grants IIS-0101134, CCR-0208595, and CCF-0430945. The research of Pierluigi San Pietro has been supported in part by MIUR grants FIRB RBAU01MCAC, COFIN 2003012437-004.
** Corresponding author (zdang@eecs.wsu.edu).

both real variables (e.g., to model time, water level, etc.) and other unbounded discrete data structures (e.g., to model the number of times a request is sent, the call stack of a recursive process, etc.). To this end, in this paper, we study real-counter automata which contain counters that take real values.

In contrast to a traditional word automaton, a two-way real-counter automaton works on a real word provided on the input tape. A real word is a bounded and closed real interval (like [0,10]), in which the two end points and a finite number of other given (intermediate) points are called markers. Each marker as well as each segment between two consecutive markers is labeled with a color drawn from a finite color set. The automaton scans through the input real word in a two-way fashion, and can distinguish whether the current read head is over a marker or within a segment. The automaton can also recognize the color of the corresponding marker/segment. During the scan, each real-counter stays unchanged or is incremented/decremented (according to the instruction that is being executed) for an amount equal to the "distance" the head moves. The automaton can also test a real-counter against 0.

In this paper, we focus on membership and emptiness problems for two-way real-counter automata ($\mathbf{R}$-2NCMs), In general, these problems are undecidable, since $\mathbf{R}$-2NCMs automata have Turing computing power. Therefore, we study some restrictions that can be applied to the model to obtain decidable membership/emptiness problems. For instance, we show decidability for $\mathbf{R}$-2NFAs (i.e., $\mathbf{R}$-2NCMs that do not have real-counters). Another restriction is reversal-boundedness: a real-counter is *reversal-bounded* (r.b. for short) if the counter changes modes between nondecreasing and nonincreasing for at most a fixed number of times during any computation. We use r.b. $\mathbf{R}$-2NCMs to denote $\mathbf{R}$-2NCMs where each real-counter is reversal-bounded. We show that the membership problem for r.b. $\mathbf{R}$-2NCMs is decidable, while the emptiness problem is undecidable. The latter undecidability remains even when the input real-words have only k markers, for a fixed k. We also study the decision problems for various versions of one-way real-counter automata. In particular, we study one-way/two-way real-counter automata that are further augmented with other unbounded discrete storages like integer-counters and/or a pushdown stack. Some of our decidability results make use of mixed linear constraints over both integer variables and real variables and the concept of mixed semilinearity over a language of real words. The concept generalizes the traditional notion of semilinearity [23] over a language of words. This makes it convenient for us to study various classes of one-way/two-way real-counter automata that have a mixed accepting condition which is a Boolean combination of mixed linear constraints over real-counters and integer-counters.

The rest of the paper is organized as follows. Section 2 defines basic notations and introduces some known results on integer-counter automata. Section 3 studies the membership and emptiness problems for two-way real-counter automata. Section 4 presents decidability results on one-way real-counter automata, also further augmented with integer-counters and a pushdown stack. Section 5 is a brief conclusion, also outlining possible applications of the real counter model to the verification of classes of hybrid systems.

2 Preliminaries

Let m and n be nonnegative integers. Consider formula $\Sigma_{1\leq i\leq m}a_ix_i + \Sigma_{1\leq j\leq n}b_jy_j \sim c$, where each x_i is a real variable, each y_j is an integer variable, each a_i, each b_j and

c are integers, $1 \leq i \leq m, 1 \leq j \leq n$, and $\sim$ is $=$, $>$, or $\equiv_d$ for some integer $d > 0$. The formula is a *mixed linear constraint* if $\sim$ is $=$ or $>$. The formula is called a *real linear constraint* if $\sim$ is $=$ or $>$ and each $b_j = 0$, $1 \leq j \leq n$. The formula is called a *discrete linear constraint* if $\sim$ is $>$ and each $a_i = 0$, $1 \leq i \leq m$. The formula is called a *discrete mod constraint*, if each $a_i = 0$, $1 \leq i \leq m$, and $\sim$ is $\equiv_d$ for some integer $d > 0$.

A formula is a *mixed* (resp. *real, Presburger*) formula if it is the result of applying quantification ($\exists$) and Boolean operations ($\neg$ and $\wedge$) over mixed (resp. real, discrete) linear constraints. It is decidable whether the formula is satisfiable. It is well-known that a Presburger formula can be written (i.e., Skolemized) as a disjunctive normal form of discrete linear constraints and discrete mod constraints. It is also known that a real formula can be written as a disjunctive normal form of real linear constraints. ¿From the results in [27], a mixed formula can also be written as a disjunctive normal form of real linear constraints, discrete linear constraints, and discrete mod constraints, when a real variable is separated into an integral part and a fractional part. We use $\mathbf{N}$ (resp. $\mathbf{R}$) to denote the set of nonnegative integers (resp. nonnegative reals). A subset S of $\mathbf{R}^m \times \mathbf{N}^n$ (resp. $\mathbf{R}^m$, $\mathbf{N}^n$) is definable by a mixed (resp. real, Presburger) formula P if S is exactly the solution set of the formula (i.e., $P(v)$ iff $v \in S$, for all v).

It is well-known that a finite automaton augmented with two integer-counters (each integer-counter can store a nonnegative integer and can independently be incremented or decremented by 1 and tested against 0), called a two-counter machine, is equivalent to a Turing machine [20]. Therefore, in order to obtain some decidable results, we need to restrict the behavior of an integer-counter. One such restriction is to make an integer-counter *reversal-bounded* [17]: there is a nonnegative integer r such that in any computation, each integer-counter can change mode between nondecreasing and nonincreasing for at most r times.

We will use the following notations: a DFA (resp. NFA) is a deterministic (resp. nondeterministic) finite automaton with a one-way input tape; a DCM (resp. NCM) is a DFA (resp. NFA) augmented with multiple integer-counters; DPDA (resp. NPDA) is a deterministic (resp. nondeterministic) pushdown automaton with a one-way input tape; DPCM (resp. NPCM) is a DPDA (resp. NPDA) augmented with multiple integer-counters. 2DFA, 2NFA, 2NCM, 2NPCM, ... will denote the variants with a two-way input tape. A two-way model is *finite-crossing* if there is a nonnegative integer k such that in any computation, the read head crosses the boundary between any two adjacent cells of the input tape no more than k times.

We use reversal-bounded NCM (resp. NPCM, 2NCM, 2NPCM, etc) to denote an NCM (resp. NPCM, 2NCM, 2NPCM, etc) where the integer-counters are reversal-bounded. Many classes of machines with reversal-bounded integer-counters have nice decidable properties (see, e.g., [17, 18, 13]), and the languages accepted by some of the one-way variants have the so-called *semilinear* property, which have been useful in showing that various verification problems concerning infinite-state systems are decidable [10, 9, 11, 14, 12, 24].

Recall the definition of semilinear sets. A set $S \subseteq \mathbf{N}^n$ is a *linear set* if there exist vectors $v_0, v_1, \ldots, v_t$ in $\mathbf{N}^n$ such that $S = \{v \mid v = v_0 + a_1 v_1 + \cdots + a_t v_t, a_i \in \mathbf{N}\}$. A set $S \subseteq \mathbf{N}^n$ is *semilinear* if it is a finite union of linear sets. It is known that S is a semilinear set if and only if it is definable by a Presburger formula. Therefore, the emptiness, containment, and equivalence problems for semilinear sets are decidable.

Let $\Sigma = \{a_1, a_2, \ldots, a_n\}$ be an alphabet. For each word w in Σ^*, define the *Parikh map* of w to be $\psi(w) = (\#_{a_1}(w), ..., \#_{a_n}(w))$, where each $\#_{a_i}(x)$ is the number of occurrences of a_i in w. For a language $L \subseteq \Sigma^*$, the *Parikh map* of L is $\psi(L) = \{\psi(w) \mid w \in L\}$. We say that a class $\mathcal{L}$ of languages over Σ is *semilinear* (or have the semilinear property) if for every language L in $\mathcal{L}$, $\psi(L)$ is a semilinear set. Many classes of languages are known to be semilinear; e.g., regular languages, context-free languages, etc. The following theorem summarizes what is known about language acceptors with reversal-bounded integer-counters.

Theorem 1. *1. Languages accepted by r.b. NCMs, r.b. NPCMs, and r.b. finite crossing 2NCMs are effectively semilinear [17]. Hence, their emptiness problem is decidable.*

2. *The emptiness problem for r.b. 2DCMs is undecidable, even when there are only two reversal-bounded integer-counters and the input comes from a bounded language (i.e., from $a_1^*...a_n^*$ for some fixed n and distinct symbols $a_1, ..., a_n$) [17].*
3. *The emptiness problem for r.b. 2DCMs with only one reversal-bounded integer-counter is decidable [18].*
4. *The emptiness problem for r.b. 2NCMs with only one reversal-bounded integer-counter and with the input coming from a bounded language is decidable [13]. (The case when the input is unrestricted is open.)*

The language acceptors mentioned so far work on words (i.e., sequences of symbols). In this paper, we will study language acceptors that work on real words, where each "symbol" is a real line segment.

3 Two-way Real-counter Automata

Let $A_0, \cdots, A_k$ be $k+1$ real numbers with $0 = A_0 < \cdots < A_k$ for some k. We use

$$W = \langle A_0, \cdots, A_k \rangle \tag{1}$$

to denote the real line between A_0 and A_k (i.e., the set $\{x : A_0 \leq x \leq A_k\}$) associated with *markers* $A_0, \cdots, A_k$. In W, A_0 (resp. A_k) is called the left (resp. right) *end marker*, and each A_i $(1 \leq i < k)$ is called an *internal marker*. Each open interval $S_i = (A_i, A_{i+1})$, $0 \leq i < k$, is called a *segment* with *length* $A_{i+1} - A_i$. Let $C = \{c_1, \cdots, c_m\}$ be a nonempty and finite set of *colors*. W is a *real word* if each segment and each marker is associated with a color in C, written

$$\langle A_0, c^0 \rangle \langle S_0, d^0 \rangle \cdots \langle A_{k-1}, c^{k-1} \rangle \langle S_{k-1}, d^{k-1} \rangle \langle A_k, c^k \rangle, \tag{2}$$

where each c^i $(0 \leq i \leq k)$ is the color of marker A_i, and each d^i $(0 \leq i < k)$ is the color of segment S_i. We use $\mathrm{color}(W) \in C^*$ to denote the sequence $c^0 d^0 \cdots c^{k-1} d^{k-1} c^k$ of colors in W. For the real word W and a color c,

- $\mathrm{marker}_c(W)$ is the number of markers in W with color c;
- $\mathrm{seg}_c(W)$ is the number of segments in W with color c;
- $\mathrm{len}_c(W)$ is the total length of segments in W with color c.

The *Parikh map* of W is the following vector $Parikh(W)$ in $\mathbf{N}^m \times \mathbf{N}^m \times \mathbf{R}^m$:

$$(\mathrm{marker}_{c_1}(W), \cdots, \mathrm{marker}_{c_m}(W), \mathrm{seg}_{c_1}(W), \cdots, \mathrm{seg}_{c_m}(W), \mathrm{len}_{c_1}(W), \cdots, \mathrm{len}_{c_m}(W))$$

A *real language* L is a set of real words. The Parikh map of L is defined to be $Parikh(L) = \{Parikh(W) : W \in L\}$. We say that L is a *mixed semilinear language* if $Parikh(L)$ is definable by a mixed formula.

Before we proceed further, some more definitions are needed. We use $\mathrm{color}(L)$ to denote the set $\{\mathrm{color}(W) : W \in L\}$. Let $\mathcal{L}$ be a family of languages, e.g., regular, context-free (accepted by NPDA), context-sensitive, 2NPDA languages, etc. L is an $\mathcal{L}$ real language if color(L) is an $\mathcal{L}$ language over alphabet C and, for any real word W, only $\mathrm{color}(W)$ decides the membership of W in L (the length of each segment in W does not matter); i.e., $W \in L$ iff $\mathrm{color}(W) \in \mathrm{color}(L)$. A *homomorphism* h is a pair of mappings $h_{\mathrm{marker}}, h_{\mathrm{seg}} : C \to C$. We use $h(W)$ to denote the real word obtained from W by modifying the color c of each marker (resp. segment) into $h_{\mathrm{marker}}(c)$ (resp. $h_{\mathrm{seg}}(c)$). We use $h(L)$ to denote the set $\{h(W) : W \in L\}$. L is *commutative* if, for any W, only $Parikh(W)$ decides the membership of W in L; i.e., $W \in L$ iff $Parikh(W) \in Parikh(L)$. A real word W is *uniform* if segments sharing the same color have the same length. We use $\mathrm{uniform}(L)$ to denote all uniform $W \in L$. The following results can be shown easily.

Theorem 2. *(1). Let $\mathcal{L}$ be a family of semilinear languages (e.g., regular, context-free, languages accepted by NFAs augmented with reversal-bounded integer-counters, etc.). Then $\mathcal{L}$ real languages are mixed semilinear languages. (2). Let h be a homomorphism. If L is a mixed semilinear language, then so is $h(L)$. (3). Let L_1 and L_2 be two mixed semilinear languages. If L_2 is commutative, then $L_1 \cap L_2$ is also a mixed semilinear language. (4). If L is a commutative and mixed semilinear language, then so is* $\mathrm{uniform}(L)$. *In fact, both of them share the same Parikh map.*

A two-way *nondeterministic finite automaton over real words* ($\mathbf{R}$-2NFA), $\mathcal{M}$, consists of a finite number of states, a two-way read head, and an input tape that stores a real word in (1). When $\mathcal{M}$ is about to make a move, it "knows" the current state. Additionally, even though $\mathcal{M}$ does not know the exact position of the head, it knows whether the head is right over a marker or is located within a segment, as well as the color of the corresponding marker or segment. A move makes use of what $\mathcal{M}$ knows and switches $\mathcal{M}$'s state, moves the head to the right or to the left (depending on the instruction of the move) for some distance and stops at a neighboring marker or within the current segment. Formally, an $\mathbf{R}$-2NFA $\mathcal{M}$ is defined as a tuple $\langle C, Q, q_0, F, T\rangle$, where C is the color set mentioned earlier, Q is the set of *states* in $\mathcal{M}$, q_0 is the *initial* state, and F is the set of accepting states. The finite set T specifies the *transitions* or *instructions* in $\mathcal{M}$, where each transition is in the form $\langle q, a, c, m, q'\rangle$, where:

- $q, q' \in Q$ indicates that, after firing the transition, the state is switched from q to q';
- $a \in \{\texttt{Marker}, \texttt{Segment}\}$ indicates whether the current position of the head is right over a marker or within a segment;
- $c \in C$ indicates the color of the corresponding marker or segment;
- $m \in \{\texttt{Left\_Marker}, \texttt{Right\_Marker}, \texttt{Left\_Segment}, \texttt{Right\_Segment}, \texttt{Stay}\}$ indicates how the head is going to move after firing the transition:
  - when $m = \texttt{Stay}$, the head does not move;
  - when $a = \texttt{Marker}$ and $m = \texttt{Left\_Segment}$ (resp. $m = \texttt{Right\_Segment}$), the head moves into the closest segment to the left (resp. to the right);
  - when $a = \texttt{Segment}$ and $m = \texttt{Left\_Segment}$ (resp. $m = \texttt{Right\_Segment}$), the head moves within the current segment to the left (resp. to the right);
  - when $a = \texttt{Segment}$ and $m = \texttt{Left\_Marker}$ (resp. $m = \texttt{Right\_Marker}$), the head moves right over the closest marker to the left (resp. to the right).

At any moment when $\mathcal{M}$ is running, if $\mathcal{M}$ tries to move beyond the left end marker or beyond the right end marker, $\mathcal{M}$ crashes.

The semantics of $\mathcal{M}$ is defined as follows. A *configuration* is a triple (W, q, x) consisting of an (input) real word W, a state q, and a nonnegative real x indicating the position of the head (i.e., the distance between the left end marker of W and the head). Clearly, when the configuration is given, one can figure out, from the values of internal markers given in W, whether the head is over a marker or located within a segment, as well as the color (also given in W) of the corresponding marker or segment. Let $t = \langle q, a, c, m, q' \rangle$ be a transition in T. The *one-step* transition relation $\stackrel{t}{\rightarrow}$ of t is defined as follows: $(W, q, x) \stackrel{t}{\rightarrow} (W, \hat{q}, \hat{x})$ iff all of the following conditions are satisfied:

- $\hat{q} = q'$;
- Suppose that W is in the form of (1) for some k. One of the following two items is true:
  - for some $0 \leq i \leq k$, $x = A_i$ (i.e., the current head is over the marker A_i). Then $a =$ `Marker` and c is exactly the color of the marker. Additionally, one of the following is true:
    - $m =$ `Stay` and $\hat{x} = x$ (i.e., the head does not move);
    - $m =$ `Left_Segment`. In this case, $i > 0$, and $A_{i-1} < \hat{x} < A_i$. That is, the current marker is not the left end marker and the new position of the head is within the segment (A_{i-1}, A_i);
    - $m =$ `Right_Segment`. In this case, $i < k$, and $A_i < \hat{x} < A_{i+1}$. That is, the new position of the head is within the segment (A_i, A_{i+1});
  - for some $0 \leq i < k$, $A_i < x < A_{i+1}$ (i.e., the current head is within segment (A_i, A_{i+1})). Then $a =$ `Segment` and c is exactly the color of the segment. Additionally, one of the following is true:
    - $m =$ `Stay` and $\hat{x} = x$ (i.e., the head does not move);
    - $m =$ `Left_Segment`. In this case, $A_i < \hat{x} < A_{i+1}$ and $\hat{x} < x$. That is, the head moves to the left but still within the same segment;
    - $m =$ `Right_Segment`. In this case, $A_i < \hat{x} < A_{i+1}$ and $\hat{x} > x$. That is, the head moves to the right but still within the same segment;
    - $m =$ `Left_Marker`. In this case, $\hat{x} = A_i$. That is, the head moves to the left marker of the segment;
    - $m =$ `Right_Marker`. In this case, $\hat{x} = A_{i+1}$. That is, the head moves to the right marker of the segment.

A *run* τ on input real word W is a sequence of one-step transitions, for some n,

$$(W, q^1, x^1) \stackrel{t^1}{\rightarrow} (W, q^2, x^2) \stackrel{t^2}{\rightarrow} \cdots \stackrel{t^{n-1}}{\rightarrow} (W, q^n, x^n).$$

The run is an *accepting run* if (W, q^1, x^1) is the initial configuration (i.e., $q^1 = q_0$ and $x^1 = A_0 = 0$) and (W, q^n, x^n) is an accepting configuration (i.e, $q^n \in F$ and the head position x^n is over the right end marker of W). W is *accepted* by $\mathcal{M}$ if $\mathcal{M}$ has an accepting run on input W. We use $L(\mathcal{M})$ to denote all the real words accepted by $\mathcal{M}$. $\mathcal{M}$ is an $\mathbf{R}$-NFA if the input tape is one-way; i.e, $\mathcal{M}$ does not move to the left during any run. One can show,

Theorem 3. *$\mathbf{R}$-2NFAs as well as $\mathbf{R}$-NFAs accept exactly regular real languages.*

Similarly, one can generalize $\mathbf{R}$-2NFAs and $\mathbf{R}$-NFAs to $\mathbf{R}$-2NPDAs and $\mathbf{R}$-NPDAs (where a pushdown stack is operated along the moves in $\mathbf{R}$-2NFAs and $\mathbf{R}$-NFAs).

Corollary 1. *$\mathbf{R}$-2NPDAs accept exactly 2NPDA real languages, and $\mathbf{R}$-NPDAs accept exactly context-free real languages.*

Remark 1. Completely in parallel to NFAs, one can show that decision problems like membership, emptiness, containment, complement, equivalence, universe, are all decidable for $\mathbf{R}$-2NFAs as well as $\mathbf{R}$-NFAs. Similarly, membership and emptiness are decidable for $\mathbf{R}$-NPDAs.

A (free) real-counter is a nonnegative real variable that can be incremented, decremented by some real amount and can be tested against zero. The counter is *reversal-bounded* if it changes mode between nondecreasing and nonincreasing for a bounded number of times. A two-way nondeterministic real-counter automaton with real input ($\mathbf{R}$-2NCM) $\mathcal{M}$ is an $\mathbf{R}$-2NFA augmented with a number of real-counters. That is, each instruction in the $\mathbf{R}$-2NFA is augmented with an enabling condition and a flow. The enabling condition compares real-counters to 0; e.g., $x_1 > 0 \wedge x_2 = 0$. The flow specifies whether a real-counter is incremented, decremented, or staying unchanged. The increment/decrement amount for each real-counter is exactly the same as the head position change after running the instruction in the $\mathbf{R}$-2NFA. $\mathcal{M}$ crashes whenever a real-counter becomes negative. Without loss of generality, we assume that when $\mathcal{M}$ accepts on an accepting state, the read head is at the right end marker and all the real-counters are zero. One can show that,

Theorem 4. *The membership problem as well as the emptiness problem for $\mathbf{R}$-2NCMs is undecidable. The undecidability remains even when the $\mathbf{R}$-2NCMs contain 2 real-counters and work on input real words with only one segment.*

Remark 2. It is open whether the membership/emptiness problems become decidable when the $\mathbf{R}$-2NCMs contain only one real-counter. This is in contrast to the fact that the membership problem for 2NCMs containing only one integer-counter is decidable while the emptiness problem is undecidable.

From Theorem 4 and Remark 2, it is necessary for us to restrict the behavior of an $\mathbf{R}$-2NCM $\mathcal{M}$ in order to obtain some decidable decision problems. One such restriction is to consider r.b. $\mathbf{R}$-2NCMs by making each real-counter in $\mathcal{M}$ reversal-bounded. We say that the input real words are *k-bounded* if they contain at most k segments. One can show,

Theorem 5. *The emptiness problem for r.b. $\mathbf{R}$-2NCMs is undecidable. The undecidability remains even when the $\mathbf{R}$-2NCMs work on k-bounded input real words for some fixed k.*

It is open whether the emptiness problem for r.b. $\mathbf{R}$-2NCMs becomes decidable when the $\mathbf{R}$-2NCMs contain only one reversal-bounded real-counter. However, we can show that the emptiness problem is decidable when the r.b. real-counter makes only one reversal.

Theorem 6. *The emptiness problem for r.b. $\mathbf{R}$-2NCMs is decidable when the $\mathbf{R}$-2NCMs contain only one reversal-bounded real-counter that makes only one reversal.*

Let L be a real language consisting of all the reals words with exactly two segments such that: the length of the first segment divided by the length of the second segment results in an integer. Clearly, L is not a mixed semilinear language. However, one can easily construct an automaton in Theorem 6 accepting L. Therefore, the automata in the theorem can accept languages that are not mixed semilinear.

Membership for $\mathbf{R}$-2NCM is decidable, while emptiness is not. We say that a r.b. $\mathbf{R}$-2NCM is with *mixed accepting condition* if at the end of an accepting run, the r.b. real-counters satisfy a given mixed formula (instead of returning to 0).

Theorem 7. *The membership problem for r.b.* $\mathbf{R}$*-2NCMs is decidable, even for r.b.* $\mathbf{R}$*-2NCMs with a mixed accepting condition.*

While by Theorem 5, emptiness is in general undecidable for r.b. $\mathbf{R}$-2NCMs, using Theorem 7, one can show that the emptiness problem for r.b. $\mathbf{R}$-2NCMs is decidable when the $\mathbf{R}$-2NCMs work on input real words with only one segment.

Theorem 8. *The emptiness problem for r.b.* $\mathbf{R}$*-2NCMs is decidable when the* $\mathbf{R}$*-2NCMs work on input real words with only one segment.*

Remark 3. According to Remark 2, we do not know whether Theorem 7 still holds when the r.b. $\mathbf{R}$-2NCMs are further augmented with a free real-counter that is not necessarily reversal-bounded.

4 One-way Real-counter Automata

The real-counter automata discussed in Section 3 are equipped with a two-way input tape. Studies in classic automata theory have shown that many decision problems become decidable when a one-way (instead of two-way) input tape is considered. In this section, we use $\mathbf{R}$-NCM to denote an $\mathbf{R}$-2NCM $\mathcal{M}$ that does not move to the left during any computation (i.e., the input tape is one-way). We first show that in general the membership/emptiness problems for $\mathbf{R}$-NCMs are undecidable.

Theorem 9. *The membership problem as well as the emptiness problem for* $\mathbf{R}$*-NCMs is undecidable. The undecidability remains even when the* $\mathbf{R}$*-NCMs contain four real-counters and work on input real words with only one segment.*

From Theorem 9, it is necessary to consider whether the emptiness problem becomes decidable when the real-counters in the $\mathbf{R}$-NCMs are reversal-bounded.

Theorem 10. *Languages accepted by r.b.* $\mathbf{R}$*-NCMs augmented with a free real-counter are mixed semilinear. Hence, the emptiness problem for the* $\mathbf{R}$*-NCMs is decidable. The decidability remains even when the* $\mathbf{R}$*-NCMs are with a mixed accepting condition.*

A $\mathbf{R}$-2NCM $\mathcal{M}$ is *finite-crossing* if there is a fixed constant k such that during any run of $\mathcal{M}$ on any input, the read head never crosses a point within the input real word for more than k times.

Theorem 11. *Languages accepted by finite-crossing r.b.* $\mathbf{R}$*-2NCMs are mixed semilinear. Hence, the emptiness problem for finite-crossing r.b.* $\mathbf{R}$*-2NCMs is decidable. The decidability remains even when the* $\mathbf{R}$*-NCMs are with a mixed accepting condition.*

Remark 4. We do not know whether Theorem 11 still holds when the $\mathbf{R}$-2NCMs are further augmented with a free real-counter. Additionally, the results in Theorems 10 and 11 become undecidable when the input real words are uniform (segments with the same color share the same length). The proof can be easily obtained following the proof of Theorem 5.

A real-counter automaton can be further augmented with unbounded discrete storage devices such as integer-counters, a pushdown stack, etc. In such an augmented automaton, instructions can be added, each of which performs a state transition and an integer-counter/stack operation while keeping other real-counters and the read head unchanged. However, decidable results are hard to obtain for two-way automata equipped with integer-counters.

Theorem 12. *(1). The emptiness problem for $\mathbf{R}$-2NFAs augmented with one integer-counter is undecidable. The undecidability therefore remains if one replaces the integer-counter with a pushdown stack. (2). The emptiness problem for $\mathbf{R}$-2NFAs augmented with two reversal-bounded integer-counters is undecidable.*

By restricting the input real word to the $\mathbf{R}$-2NFAs to be bounded, we can show:

Theorem 13. *The emptiness problem for $\mathbf{R}$-2NFAs augmented with a pushdown stack and a number of reversal-bounded integer-counters is decidable when the $\mathbf{R}$-2NFAs operate on a bounded language.*

Currently, we do not know whether the decidability remains in Theorem 13 when the $\mathbf{R}$-2NFAs are further augmented with a (reversal-bounded) real-counter. Turning to the case of one-way input, we can generalize Theorem 10 as follows.

Theorem 14. *(1). Languages accepted by r.b. $\mathbf{R}$-NCMs augmented with r.b. integer-counters are mixed semilinear. Hence, the emptiness problem for $\mathbf{R}$-NCMs is decidable. The decidability remains when the $\mathbf{R}$-NCMs are with a mixed accepting condition over the real-counters and the integer-counters. (2). The membership problem for r.b. $\mathbf{R}$-2NCMs augmented with reversal-bounded integer-counters and a pushdown stack is decidable. The decidability remains when the $\mathbf{R}$-2NCMs are with a mixed accepting condition over the real-counters and the integer-counters. (3). The emptiness problem for r.b. $\mathbf{R}$-NCMs augmented with a free real-counter, reversal-bounded integer-counters and a pushdown stack is decidable.*

Remark 5. Theorem 14(1) can be generalized to finite-crossing $\mathbf{R}$-2NCMs. Also, as in Remark 4, the results in Theorem 14 become undecidable when the input real words are uniform. The proof can also be easily followed from the proof of Theorem 5.

Theorem 14 (2) is interesting, since it entails the decidability of emptiness for $\mathbf{R}$-NFAs augmented with a free real-counter and a free integer-counter. This is in contrast to the undecidability result when the real-counter is replaced with an integer-counter.

5 Conclusions

In this paper, we introduced real-counter automata, which are two-way finite automata augmented with counters that take real values. In contrast to traditional word automata that accept sequences of symbols, real-counter automata accept real words that are bounded and closed real intervals delimited by a finite number of markers. We studied the membership and emptiness problems for one-way/two-way real-counter automata as well as those automata further augmented with other unbounded storage devices such as integer-counters and pushdown stacks. The main results are summarized in the following table.

<table>
<tr><th></th><th>r.b.$\mathbf{R}$-2NCM</th><th>r.b.$\mathbf{R}$-NCM</th><th>$\mathbf{R}$-2NFA</th><th>$\mathbf{R}$-2NCM</th><th>$\mathbf{R}$-NCM</th></tr>
<tr><td>Emp.</td><td>U (also with bounded input words). D with one-segment words or with finite-crossing or with only 1 rb counter making at most one reversal</td><td>D (also with one free real counter and a mixed accepting condition)</td><td>D. U with one free integer-counter or with 2 rb integer counters, but D on bounded languages also with a stack and rb integer counters</td><td rowspan="2">U (already with 2 free counters and one-segment words)</td><td rowspan="2">U (already with 4 free counters and one-segment words)</td></tr>
<tr><td>Mem.</td><td>D (also with a mixed accepting condition)</td><td>D also with r.b. integer counters and one pushdown stack</td><td>D</td></tr>
</table>

Table 1. Main decidability results (U=undecidable, D = decidable, Emp. = Emptiness, Mem.= Membership).

Results obtained in the previous sections can be useful in the area of formal verification, in particular for hybrid systems that contain operations on both continuous variables and discrete variables. The model itself may not be suitable for directly modeling hybrid systems. However, a different approach may be followed, by using the real counter model to study properties of hybrid systems.

Our results on real-counter automata may be used to investigate reachability problems for a class of finite-state programs augmented with real-counters. Such a program is capable of incrementing/decrementing the real-counters synchronously (for the same amount that is nondeterministically chosen) and comparing a real-counter with an integer constant. For instance, let P be a finite-state program containing a real-counter x and an integer-counter y. An instruction in P, besides its state transition, can increment/decrement x by some (nondeterministically chosen) amount. An instruction can also increment/decrement y by 1. Additionally, both x and y can be tested against a given integer constant. This program may model a discrete controller regulating one continuous, bounded physical variable (such as the water level in a reservoir) monitored only at discrete steps and measured with finite precision. The exact law governing the real variable may not be known. The difference between the measured value and the actual value may be considered as a noise, potentially disrupting system behavior.

The following reachability problem may be shown to be decidable: starting from a given state in P with both counters are 0, can P reach a designated state during which x is always bounded between two constants (e.g., 0 and 10), and when the designated state is reached, a linear constraint such as $2x - 3y + 4z > 5 \wedge 3x + y > 6z$ is satisfied? Here z denotes the total amount of increments made to the real-counter x. The decidability derives immediately from one of the results in the paper, namely the decidability of the membership problem for r.b. $\mathbf{R}$-2NCMs, with a mixed linear accepting condition, augmented with additional reversal-bounded integer-counters and a pushdown stack.

Our model of real-counter automata and the decidability results are new and are related to but disjoint with the existing results on hybrid automata. The above decidability result cannot be obtained from existing results on computing transitive closures for a restricted class of hybrid systems (e.g., [25, 5, 15, 8]). The decidability does not follow from decidable models of hybrid automata [1, 16] either. For instance, some decidable results exist for restricted hybrid automata (see, e.g., timed automata [2], some multirate automata [1, 21], initialized rectangular automata [22], etc.). However, modeling the amount z in the above reachability problem (recall that the z stands for the total amount of increments made to the real-counter x) would require a stop-watch like variable. But even under a simple set-up, it is known that timed automata augmented with one stop-watch [22] is already undecidable. The decidable reachability cannot be derived from our recent results [28] on a different model, called dense-counter machines, of real-counter programs. In a dense-counter machine, each counter can be incremented/decremented by 1 or some amount between 0 and 1. Additionally, the counter can be tested against 0. The main result in [28] shows a decidable case of a dense-counter machine, which is not strong enough to show the decidable reachability in the above mentioned example: the real-counter x in the example can be compared to an integer constant (while in a dense-counter machine, only comparisons to 0 are possible), and moreover the integer-counter y is not allowed in a dense-counter machine.

Future work will be devoted to better understanding the applicability of real-counter automata, and their relation with more established timed models, such as the timed languages of [4] (also in the two-way version of [3]) and the data languages of [6], and with decidable classes of hybrid systems, such as the Integration Graphs of [19].

References

1. R. Alur, C. Courcoubetis, T. A. Henzinger, and P. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer, 1992.
2. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994.
3. R. Alur and T. A. Henzinger. Back to the future: Towards a theory of timed regular languages. In *Proceedings of FOCS'92*. IEEE press.
4. E. Asarin, P. Caspi, and O. Maler. Timed regular expressions. *Journal of the ACM*, 49(2):172–206,.
5. B. Boigelot and P. Wolper. Symbolic verification with periodic sets. In *Proceedings of the 6th International Conference on Computer Aided Verification*, volume 818 of *Lecture Notes in Computer Science*, pages 55–67. Springer-Verlag, 1994.
6. P. Bouyer, A. Petit, and D. Therien. An algebraic characterization of data and timed languages. In *CONCUR'01*, volume 2154 of *Lecture Notes in Computer Science*, pages 248–261. Springer-Verlag, 2001.
7. E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, April 1986.
8. H. Comon and Y. Jurski. Multiple counters automata, safety analysis and Presburger arithmetic. In *CAV'98*, volume 1427 of *Lecture Notes in Computer Science*, pages 268–279. Springer, 1998.
9. Z. Dang. Pushdown time automata: a binary reachability characterization and safety verification. *Theoretical Computer Science*, 302:93–121, 2003.
10. Z. Dang, O. H. Ibarra, T. Bultan, R. A. Kemmerer, and J. Su. Binary reachability analysis of discrete pushdown timed automata. In *Proceedings of the International Conference on Computer Aided Verification (CAV 2000)*, volume 1855 of *Lecture Notes in Computer Science*, pages 69–84. Springer, 2000.
11. Z. Dang, O. H. Ibarra, and R. A. Kemmerer. Generalized discrete timed automata: decidable approximations for safety verification. *Theoretical Computer Science*, 296:59–74, 2003.

12. Z. Dang, O. H. Ibarra, and P. San Pietro. Liveness Verification of Reversal-bounded Multicounter Machines with a Free Counter. In *Proceedings of the 20th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2001)*, volume 2245 of *Lecture Notes in Computer Science*, pages 132–143. Springer, 2001.
13. Z. Dang, O. H. Ibarra, and Z. Sun. On the emptiness problems for two-way nondeterministic finite automata with one reversal-bounded counter. In *Proceedings of the 13th International Symposium on Algorithms and Computation (ISAAC 2002)*, volume 2518 of *Lecture Notes in Computer Science*, pages 103–114. Springer, 2002.
14. Z. Dang, P. San Pietro, and R. A. Kemmerer. Presburger liveness verification for discrete timed automata. *Theoretical Computer Science*, 299:413–438, 2003.
15. L. Fribourg and H. Olsen. A decompositional approach for computing least fixed-points of Datalog programs with Z-counters. *Constraints*, 2(3/4):305–335, 1997.
16. T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HYTECH: A Model Checker for Hybrid Systems. In O. Grumberg, editor, *Proceedings of the 9th International Conference on Computer Aided Verification*, volume 1254 of *Lecture Notes in Computer Science*, pages 460–463. Springer-Verlag, 1997.
17. O. H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the ACM*, 25(1):116–133, January 1978.
18. 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.
19. Y. Keste, A. Pnueli, J. Sifakis, and S. Yovine. Integration graphs: A class of decidable hybrid systems. In *Workshop on Theory of Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 179–208. Springer-Verlag, 1992.
20. 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.
21. X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An approach to the description and analysis of hybrid systems. In *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 149–178. Springer, 1992.
22. A. Puri P. Kopke, T. Henzinger and P. Varaiya. What's decidable about hybrid automata? *27th Annual ACM Symposium on Theory of Computing (STOC'95)*, pages 372–382, 1995.
23. R. Parikh. On context-free languages. *Journal of the ACM*, 13:570–581, 1966.
24. P. San Pietro and Z. Dang. Automatic verification of multi-queue discrete timed automata. In *Proceedings of the 9th Annual International Computing and Combinatorics Conference (COCOON 2003)*, volume 2697 of *Lecture Notes in Computer Science*, pages 159–171. Springer, 2003.
25. P. Z. Revesz. A closed form for datalog queries with integer order. volume 470 of *Lecture Notes in Computer Science*, pages 187–201. Springer, 1990.
26. M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Proceedings 1st Annual IEEE Symp. on Logic in Computer Science, LICS'86, Cambridge, MA, USA, 16–18 June 1986*, pages 332–344, Washington, DC, 1986. IEEE Computer Society Press.
27. V. Weispfenning. Mixed real-integer linear quantifier elimination. *Proc. Intl. Symp. on Symbolic and Algebraic Computation*, pages 129–136, Vancouver, B.C., Canada, July 29-31, 1999.
28. G. Xie, Z. Dang, O. H. Ibarra, and P. San Pietro. Dense counter machines and verification problems. In *Proceedings of the 15th International Conference on Computer Aided Verification (CAV 2003)*, volume 2759 of *Lecture Notes in Computer Science*, pages 163–175. Springer, 2003.