Dense Counter Machines and Verification Problems

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

[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 generalize the traditional definition of a multicounter machine (where the counters, which can only assume nonnegative integer values, can be incremented/decremented by 1 and tested for zero) by allowing the machine the additional ability to increment/decrement the counters by a nondeterministically chosen fractional amount δ between 0 and 1 (the δ may be different at each step). We show that, under some restrictions on counter behavior, the binary reachability set of such a machine is definable in the additive theory of the reals and integers. There are applications of this result in verification, and we give one example in the paper. We also extend the notion of "semilinear language" to "dense semilinear language" and show its connection to a restricted class of dense multicounter automata.

Keywords: infinite-state system, reachability, dense counters, additive theory of reals and integers
Category: A. Regular papers
Contact author: Zhe Dang
Email: `zdang@eecs.wsu.edu`
Tel: 509-335-7238.
Fax: 509-335-3818.

1 Introduction

In traditional automata theory, unbounded discrete data structures (e.g., integer counters, stacks, queues, etc.) are well-studied. Almost every area in computer science, including verification, has benefited from the fundamental results obtained by automata theorists. Still, practitioners in formal specification/verification keep challenging automata theorists with new models emerging in 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 systems containing both dense counters and other unbounded discrete data structures.

Dense counters are necessary in modeling many control systems and real-time applications. For instance, a dense counter may be needed to count the accumulated execution time of a number (possibly unbounded) of instances of a task. Sometimes, dense counters participate in the dynamics of a system along with other discrete counters (e.g., to record the number of times a kind of task is scheduled, or to count the total integral "price" associated with each stay at a node in a timed automaton [16]).

The importance of dense counters has already been noticed in model-checking. One of the focuses during the past ten years is the study of hybrid automata [2, 13] where dense counters follow some complex continuous flow, typically characterized by differential equations, and interact with bounded discrete control variables. In particular, timed automata [3] are a sort of hybrid automata where the clocks flow at rate 1 and can be reset to 0. Decidable results on reachability has been obtained for many forms of hybrid automata (e.g., multirate automata [2, 18], initialized rectangular automata [19]). On the other hand, linear hybrid automata in general are undecidable for reachability [19]. In fact, the undecidability remains even for timed automata augmented with one stop watch [19].

Another focus approaches dense counters in a different way. Ultimately, in the view of an external observer, the evolution of a system containing dense counters in $\mathbf{X}$ can be characterized by a formula $T(s, \mathbf{X}, s', \mathbf{X}')$, whose meaning is roughly: the system transits from control state s to s' while changing the counters from $\mathbf{X}$ to $\mathbf{X}'$. In other words, T characterizes a one-step transition of the system, called a dense counter system. This description of the system hides the details of the dynamics of the dense counters while moving from $\mathbf{X}$ to $\mathbf{X}'$. A fundamental question is then: what kinds of T would make the transitive closure T^* computable? Many model-checking queries (e.g., reachability) relies on the answer. For instance, it is known [6] that, using a complex technique for analyzing shortest paths, the transitive closure over one (non-nested) loop of T is expressible in the additive theory of reals and integers, as long as each one-step transition is in the form of $y \sim z + c$, where y, z is a dense counter or its primed form. This result is later used to show a reachability relation characterization for timed automata [7]. Some other results studying special forms of T and its (restricted) transitive closure can be found in e.g., [20, 5, 12].

Recently, it has been shown that some fundamental results in automata theory (such as [14, 15, 8]) on some restricted discrete counter machines are quite useful in studying various model-checking problems for infinite state systems containing discrete counters (e.g., [9, 11] etc.). Following this line, we believe that developing a fundamental theory (which does not exist) in the view of automata theory for these machines but with dense counters should be equally useful for studying dense counter systems. This automata-theoretic approach is different from those taken in [6, 20, 5, 12] mentioned earlier. As a starting point, in this paper we will develop a preliminary automata theory for dense counter machines.

We define a dense counter machine $\mathcal{M}$ as a finite state machine augmented with a number of dense counters. The counters can assume nonnegative real values. At each step, each counter can be incremented/decremented by 0, 1 or a nondeterministically chosen $0 < \delta < 1$. Note that δ may be different at each step. We assume w.l.o.g that for a given step, the *same* δ is used for all the counters. (This assumption can be removed.) This latter δ-increment/decrement is the essential difference between dense and discrete counters. The machine $\mathcal{M}$ can also test a counter against 0 ($= 0?, > 0?$). Since counters are assumed nonnegative, $\mathcal{M}$ crashes if a counter falls below 0. (Note that since the counters can be tested against 0, the system can actually check if it will crash and if so enter a distinguished state. So the assumption of "crashing" can be made w.l.o.g.) The following is an example instruction in $\mathcal{M}$:

s: `if` $x_1 > 0$ `then for some` $0 < \delta < 1$,
$x'_1 = x_1 + \delta,\ x'_2 = x_2 - \delta,\ x'_3 = x_3 + 1,\ x'_4 = x_4 - \delta,\ x'_5 = x_5 - \delta,$
`goto` s'.

Given two designated states s_{init} and s_{final} in $\mathcal{M}$, we study the possibility of computing the transitive closure "$\leadsto$", called the binary reachability, of $\mathcal{M}$: $\mathbf{X} \leadsto \mathbf{X}'$ iff $\mathbf{X}$ at s_{init} reaches $\mathbf{X}'$ at s_{final} in $\mathcal{M}$. On the negative side, even the state reachability problem (whether s_{init} reaches s_{final}, or equivalently, whether $\leadsto$ is empty) is in general undecidable. This is because a two (discrete) counter machine can be simulated by $\mathcal{M}$ in which no δ-increments/decrements are made. What if $\mathcal{M}$ only performs δ-increments/decrements (along with tests against 0)? In this case, the undecidability remains even when there are only four dense counters. Interestingly, still in this case, the state reachability problem becomes decidable when there are two counters. The case for three counters is open.

On the positive side, we will show some restricted versions of $\mathcal{M}$ whose binary reachability is definable in the additive theory of the reals and integers. The theory is decidable, for instance, by the Büchi-automata based decision procedure presented in [4] and a quantifier elimination technique in [21]. This will allow us to automatically verify some safety properties for the restricted $\mathcal{M}$:

Are there $\mathbf{X}$ and $\mathbf{X}'$ such that $I(\mathbf{X}, \mathbf{X}') \wedge \mathbf{X} \leadsto \mathbf{X}'$?

where I is definable in the additive theory of the reals and integers (e.g., $x_1 + x'_1 - 3x_2 > 4 \wedge x_3 - x'_2$ *is an integer* $\wedge x'_2 - x_1 < 7$). Furthermore, some liveness properties (e.g., infinitely ofteness [11]) can also be automatically verified:

Are there $\mathbf{X}^0, \cdots, \mathbf{X}^n, \cdots$ such that $\mathbf{X}^0 \leadsto \mathbf{X}^1 \leadsto \cdots \leadsto \mathbf{X}^n \leadsto \cdots$, $I(\mathbf{X}_0)$ holds, and, there are infinitely many n for which $P(\mathbf{X}_n)$ holds?

where I and P are definable in the additive theory of the reals and integers.

In this paper, we first consider a restricted version of $\mathcal{M}$ in which all the dense counters are monotonic (i.e., never decreasing). Then we consider the case when all the dense counters are reversal-bounded (the alternations between nondecreasing and nonincreasing are bounded by a fixed integer, for each counter). The concept of "reversal-boundedness" is borrowed from its counterpart for discrete counters [14]. It is not too difficult to prove that in these cases $\leadsto$ is definable in the additive theory of the reals and integers. The proof becomes more difficult when $\mathcal{M}$ is further augmented with an unrestricted dense counter (called a free counter). In the rather complex proof, we use a "shifting" technique to simplify the behavior of the free counter.

The proofs can be easily generalized to the cases when counters in $\mathcal{M}$ are of multirate. That is, a nonnegative integer rate vector $(r_1, \cdots, r_k)$ is now associated with a counter instruction such that, for each counter x_i ($1 \le i \le k$), an increment/decrement by 1 now corresponds to an increment/decrement by r_i; an increment/decrement by δ now corresponds to an increment/decrement by $r_i \cdot \delta$. For instance, the example instruction mentioned previously has the following semantics when, e.g., the rate vector is $(3, 4, 5, 6, 7)$ for $(x_1, x_2, x_3, x_4, x_5)$:

s: `if` $x_1 > 0$ `then for some` $0 < \delta < 1$,
$x'_1 = x_1 + 3 \cdot \delta,\ x'_2 = x_2 - 4 \cdot \delta,\ x'_3 = x_3 + 5 \cdot 1,\ x'_4 = x_4 - 6 \cdot \delta,\ x'_5 = x_5 - 7 \cdot \delta,$
`goto` s'.

Also note that, in an instruction, the δ need not necessarily be the same for each counter. For instance, one may modify the above instruction by associating δ_1 with x_1, x_2 and δ_2 with x_4, x_5 as follows:

s: `if` $x_1 > 0$ `then for some` $0 < \delta_1, \delta_2 < 1$,
$x'_1 = x_1 + 3 \cdot \delta_1,\ x'_2 = x_2 - 4 \cdot \delta_1,\ x'_3 = x_3 + 5 \cdot 1,\ x'_4 = x_4 - 6 \cdot \delta_2,\ x'_5 = x_5 - 7 \cdot \delta_2,$
`goto` s'.

This is because this type of instruction can always be simulated by the old ones, by adding more control states.

The automata-theoretic approach taken in this paper is new and the decidable results for the transitive closure computations on the counter systems are incomparable to previously known results (e.g., [6, 20, 5, 12]). The models, e.g., the reversal-bounded dense counter machines with a free counter, are incomparable with some restricted and decidable models for hybrid systems (e.g., timed automata [3], initialized rectangle automata [19], etc). In the future, we plan to use the results in this paper as a fundamental tool to reason about a broader class of dense counter systems. For instance, we may study a subclass of timed automata whose accumulated delays [1] are decidable over a formula of the additive theory of reals and integers. Our models are convenient in modeling accumulated delays which are simply monotonic dense counters. We can also use the model to specify some interesting systems with multiple dense/discrete counters.

Consider, e.g., the following version of the traditional producer-consumer system. A system may *produce*, *consume* or be *idle*. When in state $produce$, a resource is produced, which may be stored and later consumed while in state $consume$. The system may alternate between production and consumption states, but it may not produce and consume at the same time. Production may be stored, and used up much later (or not used at all). The novelty is that the resource may be a real number, representing for instance the available amount of a physical quantity (e.g., fuel, water or time).

This system may be easily modeled by a finite state machine with a free purely-dense counter, where the resource is added when produced or subtracted when consumed. Since the free counter may never decrease below zero (e.g., it can tested against 0), the system clearly implements the physical constraint that consumption must never exceed production. Using a dense-counter, there is an underlying assumption that a continuous variable, such as our resource, changes in discrete steps only; however, this is acceptable in many cases where a variable actually changes continuously, since the increments/decrements may be arbitrarily small.

We note that decidable versions of timed or hybrid automata do not appear to be able to specify the producer-consumer example above (even though we cannot rule out that some other decidable model can). In timed automata, when the resource is interpreted as time, the amount of production (respectively, consumption) is equivalent to the "total accumulated time spent while in state $produce$ (respectively, $consume$)". Implementing the constraint $production - consumption \geq 0$ is then tantamount to implementing a stop watch, which is known to lead to undecidability of the model. In hybrid automata, a continuous variable for $production - consumption$ must have different flow rates in state $produce$ (nonnegative) and in state $consume$ (nonpositive). A rate change, at least for the decidable class of initialized rectangular automata, forces the variable to be reinitialized to a predefined value after a state change: the value a variable had in state $produce$ is lost in state $consume$. Adding more variables does not appear to solve the problem, either, since no variable comparison is possible.

The dense-counter model of the example can be easily adapted to more complex situations. For instance, suppose that a constraint on the system is that total production is at most twice the amount of consumption. This can be described by adding two reversal-bounded dense counters, namely p, c, to store the total production (p) and twice the total consumption (c). For the sake of clarity, a nondeterministic version of the producer-consumer model $\mathcal{M}$ is shown in Figure 1, in which counter *count* denotes the difference between the total production and the total consumption. Each arc has a boolean guard and may specify an increment or decrement that is a multiple of a δ, $0 < \delta < 1$, for each counter *count*, p, c. Recall that δ may be different at eah step. If no variation is specified, then the counter stays.

When in the (initial) state *idle*, $\mathcal{M}$ may start producing or, nondeterministically, may go into the checking state if $p > 0$. While producing (in state *produce*), counter p is increased together with *count* (hence, of the same amount). Eventually consumption starts: counter c is increased in state *consume* of the double amount used to decrease $count$. Then $\mathcal{M}$ may go back to the *idle* state. Notice that $count$ may never go below 0, otherwise $\mathcal{M}$ crashes (alternatively, *count* could be tested against 0, but crashing is ruled out since it leads anyway to a nonaccepting path).

When finished producing and consuming, $\mathcal{M}$ goes into state *check*, where it decrements both p and c until p goes to 0. At this time, it moves to the final state *end*. Hence, if the value of c is not greater or equal to p, $\mathcal{M}$ crashes. Nondeterminism allows the automaton to guess a sequence of decrements δ that leads to $p = 0$ and hence to the final state *end*. To verify that the system really produces at most twice than

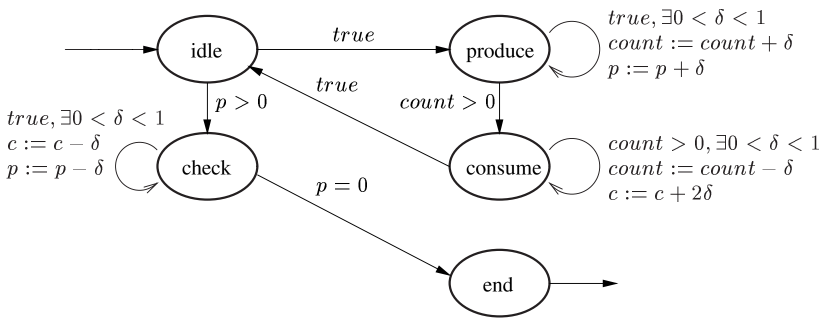

Fig. 1. A producer-consumer system $\mathcal{M}$.

it consumes, it is enough to check whether $\mathcal{M}$ may reach state *end*. Similar linear bounds, such as "total production should not exceed consumption more than 5 per cent", can be dealt with analogously.

A more sophisticated verification capability is given by the fact that the binary reachability $\leadsto^{\mathcal{M}}$ is not only computable but may be represented with a formula in the additive theory of reals and integers (hence, decidable). For instance, consider the property that the system may be implemented with finite storage (i.e., *count* is bounded). This can be expressed by the following decidable formula:

$$\exists y > 0\, \forall s \forall \mathbf{X} \left((idle, \mathbf{0}) \leadsto^{\mathcal{M}} (s, X) \;\Rightarrow\; \mathbf{X}(count) \leq y \right)$$

whose meaning is that there is a bound $y > 0$ such that, starting in $idle$ state with all counters equal to 0, $\mathcal{M}$ can only reach configurations $(s, \mathbf{X})$ where the value of *count* in $\mathbf{X}$, $\mathbf{X}(count)$, is not greater than y.

In the simple case of Figure 1, the property is obviously violated since for instance production in state *produce* may go on unbounded without consumption.

Some of the proofs can be found in the Appendix, which may be read by the PC members at their discretion.

2 Preliminaries

A dense counter is a nonnegative real variable. A dense multicounter machine $\mathcal{M}$ is a finite state machine augmented with a number of dense counters. On a move from one state to another, $\mathcal{M}$ can add an amount of $0, +1, -1, +\delta$, or $-\delta$ for some nondeterministically chosen δ, with $0 < \delta < 1$. On a move, $\mathcal{M}$ can also test a counter against 0 while leaving all the counters unchanged. Formally, we have the following definitions. A dense counter changes according to one of the following five *modes*: stay, unit increment, unit decrement, fractional increment, fractional decrement. We use $\mathbf{X}$ to denote a vector of values for dense counters $x_1, \cdots, x_k$. In the sequel, we shall use $\mathbf{X}(x_i)$ to denote the component for x_i in $\mathbf{X}$. A *mode vector* is a k-tuple of modes. There are 5^k different mode vectors. Given a mode vector $\mathbf{m} = (m_1, \cdots, m_k)$ and an amount $0 < \delta < 1$, we define $R_{\mathbf{m},\delta}(\mathbf{X}, \mathbf{X}')$ as follows: $R_{\mathbf{m},\delta}(\mathbf{X}, \mathbf{X}')$ iff for each $1 \leq i \leq k$, $\mathbf{X}(x_i) \geq 0$ and $\mathbf{X}'(x_i) \geq 0$, and each of the followings holds:

- if m_i is stay, then $\mathbf{X}'(x_i) = \mathbf{X}(x_i)$;
- if m_i is unit increment, then $\mathbf{X}'(x_i) = \mathbf{X}(x_i) + 1$;
- if m_i is unit decrement, then $\mathbf{X}'(x_i) = \mathbf{X}(x_i) - 1$;
- if m_i is fractional increment, then $\mathbf{X}'(x_i) = \mathbf{X}(x_i) + \delta$;
- if m_i is fractional decrement, then $\mathbf{X}'(x_i) = \mathbf{X}(x_i) - \delta$.

We use $R_{\mathbf{m}}(\mathbf{X}, \mathbf{X}')$ to characterize the relationship between the old values $\mathbf{X}$ and the new values $\mathbf{X}'$ under the mode vector; i.e., $R_{\mathbf{m}}(\mathbf{X}, \mathbf{X}')$ is true iff there exists some $0 < \delta < 1$ such that $R_{\mathbf{m},\delta}(\mathbf{X}, \mathbf{X}')$

holds. Notice that when $\mathbf{m}$ does not contain any mode of fractional increment/decrement, the amount δ is irrelevant.

Formally, a *(nondeterministic) dense counter machine* $\mathcal{M}$ is tuple

$$\langle S, \{x_1, \cdots, x_k\}, s_{\text{init}}, s_{\text{final}}, T\rangle, \tag{1}$$

where S is a finite set of *(control) states* where s_{init} and s_{final} are the initial and the final states. $x_1, \cdots, x_k$ are dense counters. T is a set of transitions. Each transition is a triple $(s, \mathbf{m}, s')$ of a source state s, a mode vector $\mathbf{m}$, and a destination state s'. A configuration of $\mathcal{M}$ is a pair $(s, \mathbf{X})$ of a state and (nonnegative) dense counter values. We write $(s, \mathbf{X}) \xrightarrow{\mathbf{m}} (s', \mathbf{X}')$, called a move of $\mathcal{M}$, if $(s, \mathbf{m}, s') \in T$ and $R_{\mathbf{m}}(\mathbf{X}, \mathbf{X}')$. As usual, $(s, \mathbf{X})$ *reaches* $(s', \mathbf{X}')$, written $(s, \mathbf{X}) \leadsto (s', \mathbf{X}')$, if there are states $s_0 = s, s_1, \cdots s_n = s'$, mode vectors $\mathbf{m}_1, \cdots, \mathbf{m}_n$, counter values $\mathbf{X}^0 = \mathbf{X}, \mathbf{X}^1, \cdots, \mathbf{X}^n = \mathbf{X}'$, for some n, such that

$$(s_0, \mathbf{X}^0) \overset{\mathbf{m}_1}{\to} (s_1, \mathbf{X}^1) \cdots \overset{\mathbf{m}_n}{\to} (s_n, \mathbf{X}^n). \tag{2}$$

The set of all pairs $(\mathbf{X}, \mathbf{X}')$ satisfying $(s_{\text{init}}, \mathbf{X}) \leadsto (s_{\text{final}}, \mathbf{X}')$ is called the *binary reachability* of $\mathcal{M}$. $\mathcal{M}$ can be thought of as a transducer, which is able to generate $\mathbf{X}'$ from $\mathbf{X}$ whenever $(\mathbf{X}, \mathbf{X}')$ is in the binary reachability.

Before proceeding further, some more definitions are needed. $\mathcal{M}$ is *monotonic* (resp. *purely dense*, *purely discrete*) if, in every transition $(s, \mathbf{m}, s')$ in T, the mode vector $\mathbf{m}$ does not contain any mode of unit/fractional decrement (resp. unit increment/decrement, fractional increment/decrement). Let r be a nonnegative integer. A sequence of modes is *r-reversal-bounded* if, on the sequence, the mode changes from a unit/fractional increment (followed by 0 or more stay modes) to a unit/fractional decrement and vice versa for at most r number of times. On an execution in (2) from s to s', counter x_i is *r-reversal-bounded* if the sequence of modes for x_i is r-reversal-bounded. Counter x_i is *reversal-bounded* in $\mathcal{M}$ if there is an r such that, on every execution from s_{init} to s_{final}, x_i is r-reversal-bounded. $\mathcal{M}$ is a reversal-bounded dense multicounter machine if every counter in $\mathcal{M}$ is reversal-bounded. $\mathcal{M}$ is a reversal-bounded dense multicounter machine with a free counter if all but one counters in $\mathcal{M}$ are reversal-bounded. The notion of reversal-boundedness is generalized from the same notion for discrete counters [14]. Also notice that a dense counter in $\mathcal{M}$ can be effectively restricted to be reversal-bounded, since one may use additional control states to remember each reversal and not to go over the bound. A discrete multicounter machine is a dense multicounter machine that is purely discrete and whose counters start from nonnegative integer values. Similarly, one can define a monotonic discrete multicounter machine, a reversal-bounded discrete multicounter machine, and a reversal-bounded discrete multicounter machine with a free discrete counter (NCMF).

Let m and n be positive integers. Consider a formula

$$\sum_{1 \le i \le m} a_i x_i + \sum_{1 \le j \le n} b_j y_j \sim c,$$

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

A formula is *definable in the additive theory of reals and integers (resp. reals, integers)* if it is the result of applying quantification ($\exists$) and Boolean operations ($\neg$ and $\wedge$) over mixed linear constraints (resp. dense linear constraints, discrete linear constraints); the formula is called a *mixed formula (resp. dense formula, Presburger formula)*. It is decidable whether a mixed formula is satisfiable (see [21] for a quantifier elimination procedure for mixed formulas).

Theorem 1. *The satisfiability of mixed formulas (and hence of dense formulas and Presburger formulas) is decidable.*

A set of nonnegative real/integer tuples is definable by a mixed formula with free variables: a tuple is in the set iff the tuple satisfies the formula. Similarly, a set of nonnegative real (resp. integer) tuples is definable by a dense (resp. Presburger) formula if a tuple is in the set iff the tuple satisfies the formula. One may have already noticed that, for the purpose of this paper, our definition of mixed/Presburger/dense formulas is on nonnegative variables.

Consider a mixed formula $R(x_1, \cdots, x_n, y_1, \cdots, y_m)$. By separating each dense variable x_i into a discrete variable $\lceil x_i \rceil$ for the integral part and a dense variable $\lfloor x_i \rfloor$ for the fractional part, $R(x_1, \cdots, x_n, y_1, \cdots, y_m)$ can always be written into the following form (after quantifier elimination), for some l: $R_1 \vee \cdots \vee R_l$, where each R_i is in the form of

$\exists z_1, \cdots, z_n, t_1, \cdots, t_n.\ x_1 = z_1 + t_1 \wedge \cdots \wedge x_n = z_n + t_n \wedge$

$$P(z_1, \cdots, z_n, y_1, \cdots, y_m) \wedge Q(t_1, \cdots, t_n), \tag{3}$$

where $z_1, \cdots, z_n$ are (nonnegative) discrete variables, and $t_1, \cdots, t_n$ are dense variables (defined on the interval $[0, 1)$). This representation can be easily obtained from [21]. In the representation, P is a conjunction of discrete linear constraints and discrete mod constraints and Q is a conjunction of dense linear constraints. For the given R, we define $J(N, X_1, \cdots, X_n, Y_1, \cdots, Y_m)$ if there are $x_1^i, \cdots, x_n^i, y_1^i, \cdots, y_m^i$, for all $1 \le i \le N$, such that for all $1 \le i \le N$, $R(x_1^i, \cdots, x_n^i, y_1^i, \cdots, y_m^i)$ and:

$$X_1 = \sum_{1 \le i \le N} x_1^i, \cdots, X_n = \sum_{1 \le i \le N} x_n^i, Y_1 = \sum_{1 \le i \le N} y_1^i, \cdots, Y_m = \sum_{1 \le i \le N} y_m^i.$$

Notice that all the above variables are nonnegative. Sometimes, we use $\sum R$ to denote J.

Lemma 1. *$\sum R$ is definable by a mixed formula.*

Proof. We only show two special cases; the general case can be established using (3).
Special Case 1. $R(y_1, \cdots, y_m)$ is a conjunction of discrete linear/mod constraints. Hence R is Presburger. It is known [14] that, from R, one can construct a reversal-bounded discrete multicounter machine $\mathcal{M}$ whose discrete counters are nondecreasing. $\mathcal{M}$ starts from a designated initial state with all the counters equal to 0 and then ends with a designated final state with some tuple of discrete counter values (i.e., the tuple is *reachable*). $\mathcal{M}$ has the property of "generating" R: a tuple is reachable iff it satisfies R. Clearly, $\sum R$ can also be generated by a $\mathcal{M}'$ by repeatedly executing $\mathcal{M}$ and using an additional monotonic discrete counter to remember the repetitions. Hence, $\sum R$ is Presburger.
Special Case 2. $R(x_1, \cdots, x_n)$ is a conjunction of dense linear constraints, where the dense variables are defined on $[0, 1)$. Observe that $J(N, X_1, \cdots, X_n)$ is equivalent to

$$R(\frac{X_1}{N}, \cdots, \frac{X_n}{N}) \wedge X_1 < N \cdots \wedge X_n < N.$$

By multiplying both sides of every dense linear constraint in R by N, one can show that J (i.e., $\sum R$) is definable by a mixed formula. □

Let $R_1, \cdots, R_k$ be mixed formulas over n dense variables and m discrete variables. $\mathcal{M}$ is a monotonic multicounter machine with discrete counters $y_1, \cdots, y_m$ and dense counters $x_1, \cdots, x_n$. All the counters start from 0. $\mathcal{M}$ moves from its initial state and ends with its final state. On a move from one state to another, $\mathcal{M}$ increments its counters $(x_1, \cdots, x_n, y_1, \cdots, y_m)$ by any amount $(\delta_1, \cdots, \delta_n, d_1, \cdots, d_m)$ satisfying $R_i(\delta_1, \cdots, \delta_n, d_1, \cdots, d_m)$ – here we say R_i is picked. The choice of i is given in the description of $\mathcal{M}$ and only depends on the source and destination state of the move. We define $R(X_1, \cdots, X_n, Y_1, \cdots, Y_m)$ iff $(X_1, \cdots, X_n, Y_1, \cdots, Y_m)$ are the counter values when $\mathcal{M}$ reaches the final state.

Theorem 2. *$R(X_1, \cdots, X_n, Y_1, \cdots, Y_m)$ is definable by a mixed formula.*

Proof. We use $C(N_1, \cdots, N_k)$ to denote the fact that, on a path from the initial state to the final state of $\mathcal{M}$, for each $1 \le i \le k$, N_i is the number of times when R_i is picked. Clearly, $C(N_1, \cdots, N_k)$ is Presburger [14]. The result is an application of Lemma 1. □

The following results [9] are also needed in the paper.

Theorem 3. *The binary reachability for a reversal-bounded discrete multicounter machine with a free counter is Presburger. Therefore, the same result also holds for a monotonic discrete multicounter machine and a reversal-bounded discrete multicounter machine.*

3 Decidability Results

In this section, we show a decidable characterization for various restricted versions of dense multicounter machines. The following two results can be shown using Theorem 2.

Theorem 4. *The binary reachability of a monotonic dense multicounter machine is definable by a mixed formula.*

Proof. Let $\mathcal{M}$ be a monotonic dense multicounter machine. It suffices to consider only those $\mathcal{M}$ where there is no test on the monotonic counters (since a test against 0 can be validated by keeping track of any increment made to the counter, using $\mathcal{M}$'s finite control.) The result follows immediately from the fact that it is possible to define with a mixed formula the set of tuples of net counter increments for all the counters on a path from the initial to the final state. Consider a move on the path with mode vector, say, $\mathbf{m}$. The increments made to all the counters only depends on $\mathbf{m}$, and are definable by a mixed formula after a close look at the definition of $R_{\mathbf{m}}$ in Section 2. Hence, the result follows from Theorem 2. □

Theorem 5. *The binary reachability of a reversal-bounded dense multicounter machine is definable by a mixed formula.*

Proof. Let $\mathcal{M}$ be a reversal-bounded dense multicounter machine with dense counters $x_1, \cdots, x_k$. We only consider the special case $k = 1$, since it can be easily generalized to any k. One notices that a reversal-bounded counter can be simulated by monotonic counters in the following sense. Suppose that in $\mathcal{M}$, x_1 is the reversal-bounded counter which makes one reversal (a nondecreasing phase followed by a nonincreasing phase). During an execution, x_1 may be tested against 0 many times. By properly guessing the time when x_1 becomes positive and later may become 0 (x_1 has to be nonnegative all the time by definition), one can safely assume that x_1 is tested at most once against 0 at the end of the execution. We can construct another $\mathcal{M}'$ which is exactly the same as $\mathcal{M}$ except that two monotonic counters y_1 and z_1 are used to simulate the x_1 in $\mathcal{M}$. Counter y_1 is used to record the increments made to x_1, while z_1 records the decrements made to x_1. We also have to make sure that the test on x_1 (e.g., $x_1 > 0$) at the end of the execution of $\mathcal{M}$, can also be implemented in $\mathcal{M}'$. Suppose $B'(y_1, z_1, y_1', z_1')$ is the binary reachability for $\mathcal{M}'$. The binary reachability for $\mathcal{M}$ can be expressed as $B(x_1, x_1')$ defined as $\exists y_1', z_1'\ B'(x_1, 0, y_1', z_1') \wedge\ x_1' = x_1 + y_1' - z_1' \wedge\ x_1' > 0$. When x_1 has more reversals, it can be handled similarly. Since B' is mixed linear (Theorem 4), the result follows. □

Now, we consider a dense multicounter machine $\mathcal{M}$ with k reversal-bounded counters $x_1, \cdots, x_k$ and a free counter x_0, in which s_{init} and s_{final} are two designated states. We further assume that $\mathcal{M}$ satisfies the following conditions:

- (Cond1) On any execution sequence in $\mathcal{M}$ from s_{init} to s_{final}, the reversal bounded counters are nondecreasing (i.e., never reverse),

- (Cond2) On any execution sequence in $\mathcal{M}$ from s_{init} to s_{final}, each move will change the free counter x_0 by a fractional decrement, a fractional increment, a unit decrement, or a unit increment (i.e., the free counter can not stay),
- (Cond3) On any execution sequence in $\mathcal{M}$ from s_{init} to s_{final}, the initial and ending values of the free counter are both 0, and in between, the free counter remains positive (> 0).

We first show a binary reachability characterization of $\mathcal{M}$ under the three conditions. The proof of the result is rather complex. It uses a technique of shifting the free counter values so that the free counter behavior is simplified.

Lemma 2. *The binary reachability of reversal-bounded dense multicounter machines with a free counter, satisfying the above three conditions, is definable by a mixed formula.*

In fact, the three conditions in Lemma 2 can be completely removed, using Theorem 2 and Lemma 2.

Theorem 6. *The binary reachability of reversal-bounded dense multicounter machines with a free counter is definable by a mixed formula.*

Now we generalize $\mathcal{M}$ to have *multirate* dense counters. The counters are *multirate* if a move in $\mathcal{M}$, in addition to the mode vector $\mathbf{m}$, is further associated with a k-ary positive integer vector $\mathbf{r} = (r_1, \cdots, r_k)$. The vector $\mathbf{r}$ is called a rate vector. The move can bring counter values from $\mathbf{X}$ to $\mathbf{X}'$ whenever $R_{\mathbf{m}}(\mathbf{X}, \mathbf{X}'')$, where $\mathbf{X}''$ is defined as follows, for each $1 \le i \le k$,

$$\mathbf{X}''(x_i) = \frac{\mathbf{X}'(x_i) - \mathbf{X}(x_i)}{r_i}.$$

For instance, suppose $k = 2$ and $\mathbf{m}$ is (fractional increment, fractional decrement). Let $\mathbf{r} = (4, 5)$. Then the effect of the move is to increment x_1 by 4δ and decrement x_2 by 5δ, for some $0 < \delta < 1$.

One can show that Theorem 6 (and hence Theorem 4 and Theorem 5) still holds even when the dense counters are multirate, using the following ideas. Let $\mathcal{M}$ be a reversal-bounded dense multicounter machine with a free counter. Let $x_0, x_1, \cdots, x_k$ be all the counters in $\mathcal{M}$ (x_0 is the free counter). Suppose that the reversal-bounded counters $x_1, \cdots, x_k$ are monotonic (the technique can be easily generalized). We now construct another $\mathcal{M}'$ to simulate $\mathcal{M}$ as follows. In $\mathcal{M}'$, each x_i, $1 \le i \le k$, is replaced with many monotonic counters, namely, $y^i_{\mathbf{m}}$, for all mode vectors $\mathbf{m}$. When $\mathcal{M}$ performs an instruction with mode vector $\mathbf{m}$ and a rate vector $(r_0, r_1 \cdots, r_k)$, $\mathcal{M}'$ performs the same instruction, repeated for r_0 times, on counters $(x_0, y^1_{\mathbf{m}}, \cdots, y^k_{\mathbf{m}})$ with mode vector $\mathbf{m}$ but with rate $(1, \cdots, 1)$ (i.e., single rate instead of multirate). Let $1 \le i \le k$. What is the relationship between x_i in $\mathcal{M}$ and all the $y^i_{\mathbf{m}}$ in $\mathcal{M}'$? We use δ^i to denote the net change to counter x_i after $\mathcal{M}$ performs the instruction. We use $\delta^i_{\mathbf{m}}$ to denote the net change to counter $y^i_{\mathbf{m}}$ after $\mathcal{M}'$ performs the simulated instructions. Clearly,

$$\delta^i = \frac{r_i}{r_0} \cdot \delta^i_{\mathbf{m}}.$$

Similarly, we use Δ_i to denote the net change to counter x_i on a path of $\mathcal{M}$, and use $\Delta^i_{\mathbf{m}}$ to denote the net change to counter $y^i_{\mathbf{m}}$ on the simulated path of $\mathcal{M}'$. One can show

$$\Delta_i = \frac{r_i}{r_0} \cdot \sum_{\mathbf{m}} \Delta^i_{\mathbf{m}}.$$

Since $\mathcal{M}'$ preserves the free counter x_0, and counters in $\mathcal{M}'$ is with single rate, from Theorem 6, we can show,

Theorem 7. *The binary reachability of multirate reversal-bounded dense multicounter machines (with a free counter) is definable by a mixed formula.*

So far, we have only focused on dense multicounter machines with at most one free counter. Now consider $\mathcal{M}$ with multiple free counters. Suppose that all the counters start from 0 in $\mathcal{M}$. The *state reachability problem* for $\mathcal{M}$ is: is there an execution path in $\mathcal{M}$ from the initial state to the final state? Obviously, if $\mathcal{M}$ contains two counters, the problem is undecidable. This is because, when the two dense counters only implement discrete increments/decrements, $\mathcal{M}$ is a Minsky machine [17] (a two (discrete) counter machine). A more interesting case is when $\mathcal{M}$ contains *purely dense* free counters only; i.e., the counters do not perform any discrete increments/decrements. The following theorem states that the state reachability problem is undecidable even when $\mathcal{M}$ has only four purely dense free counters.

Theorem 8. *The state reachability problem for purely dense multicounter machines is undecidable. The undecidability remains even when there are only four purely dense free counters.*

Proof. Let M be a Minsky machine with discrete counters y_1 and y_2. We construct a purely dense multicounter machine $\mathcal{M}$ with four purely dense counters z_1, z_2, x_1, x_2, and use $\mathcal{M}$ to simulate M. Then the result follows immediately. Initially, all the dense counters z_1, z_2, x_1, x_2 are 0. $\mathcal{M}$ works as a pulse generator as follows. In the first pulse, $\mathcal{M}$ makes a fractional increment to z_1 only; i.e., $z_1 := z_1 + \delta$ for some $0 < \delta < 1$. In the second pulse, $\mathcal{M}$ makes a fractional decrement to z_1 while, synchronously, makes a fractional increment to z_2. At the end of this pulse, $\mathcal{M}$ makes sure that z_1 is 0. Hence, the total effect of the pulse brings z_1 back to 0 and z_2 to δ. In the third pulse, $\mathcal{M}$ performs exactly as in the second pulse except that the roles of z_1 and z_2 are switched. The process continues by repeating the second pulse followed by the third. In each pulse, $\mathcal{M}$ is able to "generate" the same amount δ. Now, we build x_1 and x_2 into the generator. While the generator is running, whenever M increments/decrements y_i $(i = 1, 2)$ by one, $\mathcal{M}$ increments/decrements x_i $(i = 1, 2)$ by δ along a pulse. Whenever M tests y_i $(i = 1, 2)$ against 0, $\mathcal{M}$ also tests x_i against 0. □

Turning now to the case when $\mathcal{M}$ has only two purely dense free counters, we have the following decidable result. The case when there are three counters is open.

Theorem 9. *The state reachability problem for machines with only two purely dense free counters is decidable.*

Proof. Let $\mathcal{M}$ be a purely dense multicounter machine with two counters x_1, x_2. Notice that, on an execution path of $\mathcal{M}$, $\mathcal{M}$ can only make fractional increments/decrements to the counters. Below, we abstract the values of x_1, x_2 as an *abstract pair* π in the form of $(\bar{i}_1, \bar{i}_2)$ with $0 \leq i_1, i_2 \leq 2$. Intuitively, $(\bar{0}, \bar{1})$ means that x_1 is 0 and x_2 stores any "extremely small positive amount". $(\bar{1}, \bar{1})$ means that both x_1 and x_2 store the same "extremely small positive amount". $(\bar{1}, \bar{2})$ means that x_2 stores an extremely small positive amount that is larger than the positive amount stored in x_1, etc. We say that (x_1, x_2) obeys $(\bar{i}_1, \bar{i}_2)$ if the following conditions hold:

- $i_j = 0$ iff $x_j = 0, j = 1, 2$,
- $i_1 \sim i_2$ iff $x_1 \sim x_2, \sim \in \{=, <, >\}$.

In $\mathcal{M}$, a mode vector is an element (m_1, m_2), with $m_1, m_2 \in \{1, 0, -1\}$; we use "1", "0", "$-1$" to denote modes of "fractional increment", "stay", "fractional decrement", respectively. In the following, we give all the rules of updating an abstract pair on a mode vector: $(\bar{i}_1, \bar{i}_2) \xrightarrow{m_1, m_2} (\bar{i}'_1, \bar{i}'_2)$ iff, for any real $0 < r < 1$ and for any pair of nonnegative reals (x'_1, x'_2) in $[0, r) \times [0, r)$ that obeys $(\bar{i}'_1, \bar{i}'_2)$, there are nonnegative reals x_1, x_2 and δ in $[0, r)$, such that (x_1, x_2) obeys $(\bar{i}_1, \bar{i}_2)$ and

$$x'_j = x_j + m_j \cdot \delta$$

for each $j = 1, 2$. For instance, one can verify $(\bar{0}, \bar{0}) \xrightarrow{1,1} (\bar{1}, \bar{1})$, $(\bar{0}, \bar{0}) \xrightarrow{1,1} (\bar{2}, \bar{2})$, $(\bar{1}, \bar{2}) \xrightarrow{1,0} (\bar{2}, \bar{2})$, $(\bar{1}, \bar{2}) \xrightarrow{1,0} (\bar{1}, \bar{2})$, $(\bar{1}, \bar{2}) \xrightarrow{1,0} (\bar{1}, \bar{1})$, $(\bar{1}, \bar{2}) \xrightarrow{1,0} (\bar{2}, \bar{1})$, $(\bar{1}, \bar{2}) \xrightarrow{1,1} (\bar{1}, \bar{2})$, $(\bar{1}, \bar{1}) \xrightarrow{-1,0} (\bar{0}, \bar{1})$, $(\bar{1}, \bar{1}) \xrightarrow{-1,0} (\bar{0}, \bar{2})$,

$(\bar{1},\bar{1}) \xrightarrow{-1,0} (\bar{1},\bar{2})$, etc. Now, consider an execution path of $\mathcal{M}$ as below:

$$(s_0,\mathbf{X}_0) \xrightarrow{\mathbf{m}_1} (s_1,\mathbf{X}_1) \xrightarrow{\mathbf{m}_2} \cdots \xrightarrow{\mathbf{m}_n} (s_n,\mathbf{X}_n), \tag{4}$$

where $s_0 = s_{\text{init}}$, and $s_n = s_{\text{final}}$, $\mathbf{X}_0 = (0,0)$. By uniformly shrinking all the fractional increments and decrements on the path of (4) (obviously, doing this still preserves all the tests and makes the path valid), one can assume that each fractional increment/decrement is by an amount less than $r = \frac{1}{n}$. Therefore, the counter values are less than 1 during the path. Corresponding to the path in (4), we construct a sequence

$$(s_0,\pi_0) \xrightarrow{\mathbf{m}_1} (s_1,\pi_1) \xrightarrow{\mathbf{m}_2} \cdots \xrightarrow{\mathbf{m}_n} (s_n,\pi_n), \tag{5}$$

where the abstract pairs satisfy: for each $0 \le i < n$, $\pi_i \xrightarrow{\mathbf{m}_1} \pi_{i+1}$, and for each $0 \le i \le n$, $\mathbf{X}_i$ obeys π_i. It is routine to show that, once (4) is given, the desired sequence (not necessarily unique) $\pi_0,\cdots,\pi_n$ always exists. Conversely, once we are given (5), we can always find $\mathbf{X}_0,\cdots,\mathbf{X}_n$, all bounded by r, satisfying (4). The result follows immediately, since, for the purpose of state reachability, $\mathcal{M}$ can be abstracted as a finite state system. □

4 Safety/Liveness Verification

From the binary reachability characterizations given in the previous section, one can establish decidable results on various verification problems for a reversal-bounded dense multicounter machine $\mathcal{M}$ with a free counter, even when counters in $\mathcal{M}$ are of multirate.

The *binary reachability problem* for $\mathcal{M}$ is defined as follows: Given a mixed formula $I(\mathbf{X},\mathbf{X}')$, are there $\mathbf{X}$ and $\mathbf{X}'$ such that $I(\mathbf{X},\mathbf{X}')$ holds and $(s_{\text{init}},\mathbf{X})$ reaches $(s_{\text{final}},\mathbf{X}')$ in $\mathcal{M}$? An example of the problem is as follows:

> Is there $(x_1,x_2,x_3,x_1',x_2',x_3')$ satisfying "$x_1'-2x_1+x_2$ *is an integer* $\wedge x_3'+x_2' > x_3+x_2+x_1+1$" and $(s_{\text{init}},x_1,x_2,x_3)$ reaches $(s_{\text{final}},x_1',x_2',x_3')$ in $\mathcal{M}$?

Using Theorem 1 and Theorem 6, one can show that the binary reachability problem is decidable.

Additionally, one can also consider the *mixed i.o. (infinitely often) problem* for $\mathcal{M}$ as follows. Given two mixed formulas I and P, the problem is to decide whether there is an infinite execution path of $\mathcal{M}$

$$(s^0,\mathbf{X}^0) \to (s^1,\mathbf{X}^1) \to \cdots \to (s^n,\mathbf{X}^n) \to \cdots$$

such that $s^0 = s_{\text{init}}, I(\mathbf{X}_0)$ holds, and there are infinitely $n_1 < n_2 < \cdots$ such that $P(X^{n_i})$ holds for all i. An example of the problem is as follows: Is there an infinite run of $\mathcal{M}$ such that

$$x_3 > x_2 + x_1 - 1 \wedge 2x_2 - x_1 \textit{ is an integer}$$

is satisfied for infinitely many times? The problem can be used to formalize some fairness properties along an ω execution path of a transition system (see also [11]). Because of Theorem 6, $\mathcal{M}$ is a mixed linear system in the sense of [10]. From the main theorem in [10], one can conclude that the mixed i.o. problem is decidable.

A dense monotonic counter machine with a free counter (with multirate counters) can be used to model a dense counter system with counter instructions in the following form

$$\exists 0 < \delta < 1.\ x_0' = x_0 + r_0 \cdot t_0 \wedge x_1' = x_1 + r_1 \cdot t_1 \wedge \cdots \wedge x_k' = x_k + r_k \cdot t_k,$$

where, $r_0,\cdots,r_k$ are positive integers (rates), x_0 is the free counter, and $x_1,\cdots,x_k$ are the monotonic counters. We use the primed variables to indicate the new values. The term t_0 for the free counter x_0 is one of $0,\delta,-\delta,1,-1$. The term t_i, $1 \le i \le k$, for monotonic counter x_i is one of $0,\delta,1$. In addition, the counters can be tested against 0. Though the machine models are intended to be a fundamental tool to reason about a broader class of dense counter systems, using Theorem 6, one can also use the model to specify some interesting systems with multiple dense/discrete counters (e.g., the consumer/producer system in Section 1).

5 Discussions

It is a natural idea to transform a dense counter machine $\mathcal{M}$ into a dense counter automaton (i.e., acceptor) $\mathcal{A}$ in which a one-way input tape is provided. In contrast to a traditional word automaton, a dense word is provided on the input tape. We elaborate on this as follows. A *block* B is a pair (c, δ), where $c \in C$ is the color of the block and $0 \leq \delta \leq 1$ is called the block's *length*. There are only finitely many possible colors in C. Let $Blocks$ be the set of all blocks on colors in C and $Blocks^*$ be the free monoid generated by the infinite set $Blocks$. Hence, there exists an associative operation (called concatenation) of blocks with an identity ϵ. The Kleene closures * and $^+$ are defined as usual using concatenation. A *dense word* is an element of $Blocks^*$ and a *dense language* is a subset of $Blocks^*$. Given a dense word w as input, $\mathcal{A}$ reads the blocks in w one by one. On reading a block, $\mathcal{A}$ updates its control state and the counters in the same way as $\mathcal{M}$ does, except that $\mathcal{A}$ knows the color of the block, can distinguish whether the length of the block is 0, 1, or strictly greater than 0 and strictly less than 1. $\mathcal{A}$ can increment/decrement some of the counters by 0, 1, or the length of the block. In this way, one can similarly define dense monotonic counter automata, dense reversal-bounded counter automata, and dense reversal-bounded counter automata with a free counter.

For a dense word w, we use $\#_c(w)$ (resp. $l_c(w)$) to denote the total number (resp. length) of blocks with color c in w. We use $Parikh(w)$ to denote the tuple of counts $\#_c(w)$ and $l_c(w)$ for each $c \in C$; i.e., $Parikh(w) = (\#_{c_1}(w), \cdots, \#_{c_n}(w), l_{c_1}(w), \cdots, l_{c_n}(w))$. For a dense language L, we use $Parikh(L)$, the Parikh map of L, to denote the set of all $Parikh(w)$ for $w \in L$. A dense language L is a *mixed semilinear* dense language if $Parikh(L)$ is definable by a mixed formula F over n integer variables and n dense variables: For all $x_1, \cdots, x_n \in \mathbf{N}$ and for all $y_1, \cdots, y_n \in \mathbf{R}$, $(x_1, \cdots, x_n, y_1, \cdots, y_n) \in Parikh(L)$ iff $F(x_1, \cdots, x_n, y_1, \cdots, y_n)$ holds. The above definition of a Parikh map for a dense language can be seen as a natural extension of the concept of Parikh maps for discrete languages. Indeed, one can treat a discrete language $\mathcal{L}$ as a special dense language $L_{\mathcal{L}}$, where a symbol in C is understood as a unit block. One can easily verify that $\mathcal{L}$ is a semilinear discrete language iff $L_{\mathcal{L}}$ is a mixed semilinear dense language. So, over discrete languages, our definition of "mixed semilinearity" coincides with the traditional "semilinearity" definition.

It is straightforward to verify the following theorem, using the results in Section 3.

Theorem 10. *Dense languages accepted by dense reversal-bounded counter automata with a free counter are mixed semilinear languages. Hence, the emptiness and infiniteness problems for these automata are decidable.*

The dense languages in Theorem 10 can be quite complex; e.g.,

$$L = \{(c, l_1) \cdots (c, l_{2n})(d, r_1) \cdots (d, r_{2n}) : l_{2k-1} > l_{2k}, r_{2k-1} > r_{2k}, \text{for each } 1 \leq k \leq n\}.$$

One can define a deterministic dense reversal-bounded counter automaton in the usual way, with the additional requirement that the transition the automaton makes on an input block (c, δ) with $0 < \delta < 1$ should be the same if the block is replaced by any (c, δ') with $0 < \delta' < 1$.

The following corollaries are easily verified:

Corollary 1. *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be dense reversal-bounded counter automata.*

1. *We can effectively construct dense reversal-bounded counter automata $\mathcal{A}$ and $\mathcal{A}'$ accepting $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ and $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$, respectively.*
2. *IF $\mathcal{A}_1$ is deterministic, we can effectively construct a deterministic dense reversal-bounded counter automaton accepting the complement of $L(\mathcal{A}_1)$.*

If one of $\mathcal{A}_1$ or $\mathcal{A}_2$ (but not both) has a free counter, then $\mathcal{A}$ and $\mathcal{A}'$ will also have a free counter. It follows that the class of languages accepted by deterministic dense reversal-bounded counter automata is closed under the boolean operations.

Corollary 2. *1. The emptiness, infiniteness, and disjointness problems for dense reversal-bounded counter automata are decidable. 2. The containment and equivalence problems for deterministic dense reversal-bounded counter automata are decidable.*

Notice that the universe problem for dense reversal-bounded counter automata is undecidable, since the undecidability holds even for discrete counters [14]. Obviously, from Theorem 10, when the automata (even with a free counter) are deterministic, the universe problem is decidable.

References

1. R. Alur, C. Courcoubetis, and T. A. Henzinger. Computing accumulated delays in real-time systems. *Formal Methods in System Design: An International Journal*, 11(2):137–155, August 1997.
2. 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*, pages 209–229, 1992.
3. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994.
4. B. Boigelot, S. Rassart, and P. Wolper. On the expressiveness of real and integer arithmetic automata. In *ICALP'98*, volume 1443 of *Lecture Notes in Computer Science*, pages 152–163. Springer-Verlag, 1998.
5. B. Boigelot and P. Wolper. Symbolic verification with periodic sets. In *CAV'94*, volume 818 of *LNCS*, pages 55–67. Springer, 1994.
6. H. Comon and Y. Jurski. Multiple counters automata, safety analysis and Presburger arithmetic. In *CAV'98*, volume 1427 of *LNCS*, pages 268–279. Springer, 1998.
7. H. Comon and Y. Jurski. Timed automata and the theory of real numbers. In *CONCUR'99*, volume 1664 of *LNCS*, pages 242–257. Springer, 1999.
8. Z. Dang, O. H. Ibarra, and Z. Sun. On the emptiness problems for two-way nondeterministic finite automata with one reversal-bounded counter. In *ISAAC'02*, volume 2518 of *LNCS*, pages 103–114. Springer, 2002.
9. Zhe Dang, O. H. Ibarra, T. Bultan, R. A. Kemmerer, and J. Su. Binary reachability analysis of discrete pushdown timed automata. In *CAV'00*, volume 1855 of *LNCS*, pages 69–84. Springer, 2000.
10. Zhe Dang and Oscar H. Ibarra. On the existence of ω-chains for transitive mixed linear realtions and its applications. *International Journal of Foundations of Computer Science*, 13(6):911–936, 2002.
11. Zhe Dang, P. San Pietro, and R. A. Kemmerer. On Presburger Liveness of Discrete Timed Automata. In *STACS'01*, volume 2010 of *LNCS*, pages 132–143. Springer, 2001.
12. 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.
13. T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: A Model Checker for Hybrid Systems. In *CAV'97*, volume 1254 of *LNCS*, pages 460–463. Springer, 1997.
14. O. H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the ACM*, 25(1):116–133, January 1978.
15. 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.
16. K. Larsen, G. Behrmann, Ed Brinksma, A. Fehnker, T. Hune, P. Pettersson, and J. Romijn. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In *CAV'01*, volume 2102 of *LNCS*. Springer, 2001.
17. 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.
18. X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An approach to the description and analysis of hybrid systems. In *Hybrid Systems*, pages 149–178, 1992.
19. A. Puri P. Kopke, T. Henzinger and P. Varaiya. What's decidable about hybrid automata? *27th Annual ACM Symposium on Theory of Computing (STOCS)*, pages 372–382, 1995.
20. P. Z. Revesz. A closed form for datalog queries with integer order. *Lecture Notes in Computer Science*, 470:187–201, 1990.
21. 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.

Appendix: Proofs not presented in the paper

Proof of Lemma 2

Proof. Let $\mathcal{M}$ be a dense multicounter machine with k reversal-bounded counters $x_1, \cdots, x_k$ and a free counter x_0, satisfying the three conditions.

Let $\mathbf{m}$ be a mode vector. $\mathbf{m}$ is a *f-increment (resp. f-decrement, d-increment, d-decrement)* if the mode $\mathbf{m}(x_0)$ of the free counter x_0 in $\mathbf{m}$ is a fractional increment (resp. fractional decrement, discrete increment, discrete decrement). When $\mathbf{m}$ is a f-increment (resp. f-decrement), we use $Discr([\mathbf{m}])$ to denote the result of replacing each "fractional increment" with "unit increment" and each "fractional decrement" with "unit decrement" in all components in $\mathbf{m}$. Similarly, we use $Stay([\mathbf{m}])$ to denote the result of replacing each "fractional increment" and each "fractional decrement" with "stay". Clearly, both $Discr([\mathbf{m}])$ and $Stay([\mathbf{m}])$ are mode vectors. In particular, $Discr([\mathbf{m}])$ is a d-increment (resp. d-decrement) iff $\mathbf{m}$ is a f-increment (resp. f-decrement). $Stay([\mathbf{m}])$ is a mode vector whose components are all "stay".

From the given $\mathcal{M}$, one can construct a nondeterministic counter machine M by replacing, for each f-increment and f-decrement mode $\mathbf{m}$, either all the moves of mode $\mathbf{m}$ with those of mode $Discr([\mathbf{m}])$, or, all the moves of mode $\mathbf{m}$ with those of mode $Stay([\mathbf{m}])$. Such replacements will result in several such M's (the number is uniformly bounded for the given k). Notice that in each M, a move in $\mathcal{M}$ that was originally a unit increment/decrement or stay is unchanged. M designates two states as its initial and final states. Initially, all the counters start from some nonnegative integer values in the initial state and end in the final state with some other nonnegative integer values, but in between the free counter does not reach 0. There are only finitely different machines M one can construct (since there are bounded number of modes for the given k and there are bounded number of choices for the initial and final states). Clearly, each M is a NCMF with a free discrete counter y_0 and monotonic counters $y_1, \cdots, y_k$. In the sequel, we shall call such a M as a NCMF obtained from $\mathcal{M}$.

Consider an execution path p of $\mathcal{M}$ that goes from s_{init} to s_{final} as follows:

$$(s_0, \mathbf{X}_0) \stackrel{\mathbf{m}_1}{\to} (s_1, \mathbf{X}_1) \stackrel{\mathbf{m}_2}{\to} \cdots \stackrel{\mathbf{m}_n}{\to} (s_n, \mathbf{X}_n) \tag{6}$$

for some $n > 0$, where, according to the three conditions, $s_0 = s_{\text{init}}$, $s_n = s_{\text{final}}$, $\mathbf{X}_0(x_0) = \mathbf{X}_n(x_0) = 0$ (the free counter starts and ends with 0), but for each $0 < i < n$, $\mathbf{X}_i(x_0) > 0$ (in between, the free counter remains positive). For each $1 \le i \le n$, $(s_{i-1}, \mathbf{X}_{i-1}) \stackrel{\mathbf{m}_i}{\to} (s_i, \mathbf{X}_i)$ indicates a move in $\mathcal{M}$; we use δ_i to denote the change made to the free counter on the move (i.e., $\delta_i = |\mathbf{X}_i(x_0) - \mathbf{X}_{i-1}(x_0)|$). Clearly, when $\mathbf{m}_i$ is a f-increment/decrement, $0 < \delta_i < 1$. Otherwise, $\mathbf{m}_i$ is a d-increment/decrement and $\delta_i = 1$. To ease our presentation, by bringing in these δ's, we simply write

$$(\mathbf{m}_1, \delta_1); \cdots; (\mathbf{m}_n, \delta_n) \tag{7}$$

to represent (6). Let $\mathbf{m}$ be any fixed f-increment mode that appears in (7). Suppose that $\mathbf{m}_{i_1}, \cdots, \mathbf{m}_{i_b}$ are all occurrences of $\mathbf{m}$ in (7), for some $b \ge 1$. Now, we adjust the values of $\delta_{i_1}, \cdots, \delta_{i_b}$ into a sequence of 1's, followed by a fractional value and a sequence of 0's. Consequently, the sequence of modes $\mathbf{m}_{i_1}, \cdots, \mathbf{m}_{i_b}$ are also adjusted. Let $\Delta = \sum_{1 \le j \le b} \delta_{i_j}$. Clearly, $\Delta > 0$. Let $c = 1 + \lceil \Delta \rceil$. The adjustments are made as below:

- for each $1 \le j \le \lceil \Delta \rceil$, the new value of δ_{i_j} is 1, and the new value of $\mathbf{m}_{i_j}$ is $Discr(\mathbf{m}_{i_j})$;
- if $\lfloor \Delta \rfloor = 0$, then the new value of $\delta_{i_c} = 0$ and the new value of $\mathbf{m}_{i_c}$ is $Stay(\mathbf{m}_{i_c})$, else the new value of $\delta_{i_c} = \lfloor \Delta \rfloor$ and the value of $\mathbf{m}_{i_c}$ remains. The index of i_c is called a *bridge index*;
- for each $c + 1 \le j \le b$, the new value of δ_{i_j} is 0 and the new value of $\mathbf{m}_{i_j}$ is $Stay(\mathbf{m}_{i_j})$.

Similarly, when $\mathbf{m}$ is a f-decrement mode that appears in (7), the adjustments are made as below (where $c = b - \lceil \Delta \rceil$):

- for each $b - \lceil \Delta + 1 \rceil \le j \le b$, the new value of δ_{i_j} is 1, and the new value of $\mathbf{m}_{i_j}$ is $Discr(\mathbf{m}_{i_j})$;

- if $\lfloor \Delta \rfloor = 0$, then the new value of $\delta_{i_c} = 0$ and the new value of $\mathbf{m}_{i_c}$ is $Stay(\mathbf{m}_{i_c})$, else the new value of $\delta_{i_c} = \lfloor \Delta \rfloor$ and the value of $\mathbf{m}_{i_c}$ remains. Similarly, the index of i_c is also called a *bridge index*;
- for each $1 \leq j \leq c-1$, the new value of δ_{i_j} is 0 and the new value of $\mathbf{m}_{i_j}$ is $Stay(\mathbf{m}_{i_j})$.

After the adjustments made for every f-increment/decrement mode appearing in (7), we use

$$(\mathbf{m}'_1, \delta'_1); \cdots ; (\mathbf{m}'_n, \delta'_n) \tag{8}$$

to denote the result. Building these new modes and δ's into (6) to reflect the adjustments, one obtains

$$(s_0, \mathbf{X}'_0) \stackrel{\mathbf{m}'_1}{\to} (s_1, \mathbf{X}'_1) \stackrel{\mathbf{m}'_2}{\to} \cdots \stackrel{\mathbf{m}'_n}{\to} (s_n, \mathbf{X}'_n) \tag{9}$$

where $\mathbf{X}'_0 = \mathbf{X}_0$ and, clearly, $\mathbf{X}'_n = \mathbf{X}_n$. Also notice that on the new path in (9) the free counter never reaches 0 except at s_0 and s_n. Unfortunately, the path in (9) may not even be a valid path in $\mathcal{M}$ since the mode vectors in each move in (6) are changed. In the following, we will show that another machine $\mathcal{M}'$ can be constructed such that (9) is an execution path of $\mathcal{M}'$.

Suppose that, in (7), there are K number of bridge indexes (K is uniformly bounded by the number of f-increment/decrement modes). Hence, (8) can be further partitioned into $K+1$ subsequences (called segments) $p'_0, p'_1, \cdots, p'_K$ delimited by a pair on each bridge index. More precisely, let $1 \leq i_1 < \cdots < i_K \leq n$ be all the bridge indexes in (7). Define p'_j, $0 \leq j \leq K$, as follows. The first segment p'_0 is in the form of $(\mathbf{m}'_1, \delta'_1); \cdots ; (\mathbf{m}'_{i_1-1}, \delta'_{i_1-1})$. For each $1 \leq j \leq K-1$, the segment p'_j is

$$(\mathbf{m}'_{i_j+1}, \delta'_{i_1+1}); \cdots ; (\mathbf{m}'_{i_{j+1}-1}, \delta'_{i_{j+1}-1}).$$

Finally, the last segment p'_K is in the form of $(\mathbf{m}'_{i_K+1}, \delta'_{i_K+1}); \cdots ; (\mathbf{m}'_n, \delta'_n)$. Connecting with the bridges, we can write (8) into

$$p'_0; (\mathbf{m}'_{i_1}, \delta'_{i_1}); \cdots ; (\mathbf{m}'_{i_{K-1}}, \delta'_{i_{K-1}}); p'_{K-1}; (\mathbf{m}'_{i_K}, \delta'_{i_K}); p'_K. \tag{10}$$

On a segment, there are only discrete moves; i.e., moves of d-increments/decrements and stays. How did we obtain a segment? We use $p_0, \cdots, p_K$ to represent the counterparts of $p'_0, \cdots, p'_K$ in (7). A close look on the adjustment procedure reveals that each p'_j can be obtained from p_j for each f-increment/decrement mode $\mathbf{m}$ that appears in (7),

- by replacing every occurrence $\mathbf{m}$ on p_j with $Discr(\mathbf{m})$, or,
- by replacing every occurrence $\mathbf{m}$ on p_j with $Stay(\mathbf{m})$.

The replacements to obtain p'_j are completely characterized by a *replacement mapping* $\mathcal{R}_j$ that maps each f-increment/decrement mode $\mathbf{m}$ on (7) to either $Discr(\mathbf{m})$ or $Stay(\mathbf{m})$. There are only bounded number of choices for each $\mathcal{R}_j$. Furthermore, the mappings of $\mathcal{R}_j$'s also satisfy some additional properties as follows. Let $\mathbf{m}$ be any fixed f-increment/decrement mode that appears in (7). We use i_t to denote the bridge index for this mode (see the adjustment procedure); i.e., the t-th bridge on (10). Clearly, for each replacement mapping, if $\mathbf{m}$ is a f-increment, then this $\mathbf{m}$ is replaced with $Discr(\mathbf{m})$ in all $p'_0, \cdots, p'_{t-1}$, and with $Stay(\mathbf{m})$ in all $p'_t, \cdots, p'_K$; i.e.,

$$\mathcal{R}_0(\mathbf{m}) = \cdots = \mathcal{R}_{t-1}(\mathbf{m}) = Discr(\mathbf{m}) \wedge \mathcal{R}_t(\mathbf{m}) = \cdots = \mathcal{R}_K(\mathbf{m}) = Stay(\mathbf{m}). \tag{11}$$

On the other hand, if $\mathbf{m}$ is a f-decrement, then this $\mathbf{m}$ is replaced with $Stay(\mathbf{m})$ in all $p'_0, \cdots, p'_{t-1}$, and with $Discr(\mathbf{m})$ in all $p'_t, \cdots, p'_K$; i.e.,

$$\mathcal{R}_0(\mathbf{m}) = \cdots = \mathcal{R}_{t-1}(\mathbf{m}) = Stay(\mathbf{m}) \wedge \mathcal{R}_t(\mathbf{m}) = \cdots = \mathcal{R}_K(\mathbf{m}) = Discr(\mathbf{m}). \tag{12}$$

Each segment p'_j, which contains discrete moves only, can be simulated by a NCMF M_j that is obtained from $\mathcal{M}$ using replacement mapping $\mathcal{R}_j$ and starts from the starting state of p'_j and ends with the ending state of p'_j. Recall that, by definition, we require that the discrete free counter y_0 in M_j never reaches 0 during the simulation (excluding the two ends). We use $\mathbf{X}'_{i_j}(x_0)$ (by convention, $i_0 = 0$, $i_{K+1} = n+1$) to denote the starting value of the dense free counter in $\mathcal{M}$ on segment p'_j (the index i_j corresponds to the one in (10) and $\mathbf{X}'_{i_j}$ corresponds to the one in (9)). The simulation is faithful in the following sense:

(*) When the discrete free counter y_0 in M_j starts from $\lceil \mathbf{X}'_{i_j}(x_0) \rceil$, and monotonic counters $y_1, \cdots, y_k$ start from 0, M_j can finally reach the ending state with the discrete free counter taking value $\lceil \mathbf{X}'_{i_{j+1}-1}(x_0) \rceil$, and with all the monotonic counters taking values $\mathbf{X}'_{i_{j+1}-1}(x_1) - \mathbf{X}'_{i_j}(x_1), \cdots, \mathbf{X}'_{i_{j+1}-1}(x_k) - \mathbf{X}'_{i_j}(x_k)$, respectively.

We use $B_j(y_0, y_1, \cdots, y_k, y'_0, y'_1, \cdots, y'_k)$ to denote the binary reachability of NCMF M_j. From Theorem 3, B_j is Presburger. To reflect (*), we define a relation $B'_j(x_0, x_1, \cdots, x_k, x'_0, x'_1, \cdots, x'_k)$ as follows:

$$\exists y_0, y'_0, y'_1, \cdots, y'_k \in \mathbf{N}.\ B_j(y_0, 0, \cdots, 0, y'_0, y'_1, \cdots, y'_k) \wedge$$

$$\lceil x_0 \rceil = y_0 \wedge \lceil x'_0 \rceil = y'_0 \wedge y'_1 = x'_1 - x_1 \wedge \cdots \wedge y'_k = x'_k - x_k. \tag{13}$$

Clearly, B'_j is mixed linear. In this way, (*) can be reformulated as $B'_j(\mathbf{X}'_{i_j}, \mathbf{X}'_{i_{j+1}-1})$, which characterizes the reachability relation of segment p'_j. From here, according to (10) and the definition of B'_j, we can further rewrite the transition sequence in (9) into a sequential composition:

$$B'_0(\mathbf{X}'_0, X'_{i_1-1}) \wedge R_{\mathbf{m}'_{i_1}, \delta_{\mathbf{m}'_{i_1}}}(X'_{i_1-1}, X'_{i_1}) \wedge B'_1(X'_{i_1}, X'_{i_2-1}) \wedge \cdots \wedge$$

$$R_{\mathbf{m}'_{i_K}, \delta_{\mathbf{m}'_{i_K}}}(X'_{i_K-1}, X'_{i_K}) \wedge B'_K(X'_{i_K}, \mathbf{X}'_n) \tag{14}$$

conjuncted with a requirement that the dense free counter does not reach 0 in between; i.e.,

$$X'_{i_1-1}(x_0) > 0 \wedge X'_{i_1}(x_0) > 0 \wedge \cdots \wedge X'_{i_K-1}(x_0) > 0 \wedge X'_{i_K}(x_0) > 0, \tag{15}$$

where $R_{\mathbf{m}'_{i_1}, \delta_{\mathbf{m}'_{i_1}}}$, as defined before, characterizes one move with mode $\mathbf{m}'_{i_1}$. We use $B(\mathbf{X}_0, \mathbf{X}_n)$ to denote the conjunction of (14) and (15) after existentially eliminating all the other variables such as $\delta_{\mathbf{m}'_{i_1}}, \cdots, \delta_{\mathbf{m}'_{i_K}}$, and all the intermediate $X'_{i_1-1}, X'_{i_1}, \cdots$. Clearly, since K is uniformly bounded, B is mixed linear.

To sum up, we have shown that, if $\mathbf{X}_0$ reaches $\mathbf{X}_n$ as defined in (6), then $B(\mathbf{X}_0, \mathbf{X}_n)$ holds. But, this only tells a half of the story. In the following, we argue that the converse is also true: if $B(\mathbf{X}_0, \mathbf{X}_n)$ holds, then one can always find an execution path in the form of (6) in $\mathcal{M}$. Suppose that $B(\mathbf{X}_0, \mathbf{X}_n)$ is true (with, as we assumed, $\mathbf{X}_0(x_0) = \mathbf{X}_n(x_0) = 0$). One can reverse the above reasoning from (14,15) all the way back to (10), (9) and (8). The key thing now is how to adjust those δ''s in (8) to some δ's in (7) such that sequence (9) can be adjusted into a sequence in the form of (6). Fortunately, this can be easily done. On the sequence of (9), the dense free counter x_0 will not reach 0 in between. We use $h > 0$ to denote the lowest point of the counter. Let $\mathbf{m}$ be an f-increment mode. For a pair $(\mathbf{m}'_i, \delta'_i)$ in (8) and $\mathbf{m}_i = \mathbf{m}$ in (7), if $\mathbf{m}'_i$ is obtained with $\mathbf{m}'_i = Discr(\mathbf{m}_i)$, (so now $\delta'_i = 1$), we take $\delta_i = 1 - \tau$. If $\mathbf{m}'_i = Stay(\mathbf{m}_i)$, we take $\delta_i = \pi$. Finally, if $\mathbf{m}'_i = \mathbf{m}_i$ (i now is a bridge index), we take $\delta_i = \delta'_i + \eta$. How to choose the values of τ, π, η? Suppose on the sequence of (7), there are totally t number of occurrences of $\mathbf{m}$, among which t_1 occurrences are mapped to $Discr(\mathbf{m})$ in (8), t_2 occurrences are mapped to $Stay(\mathbf{m})$ in (8), and $t_3 \leq 1$ occurrences are mapped to $\mathbf{m}$ in (8). One may choose τ and π to be an extremely small positive amount, and η to be an extremely small positive/0/negative amount, such that the following holds:

$$t_1 \cdot \tau = t_2 \cdot \pi + t_3 \cdot \eta.$$

This choice is to make sure that the net change of the dense free counter with respect to mode $\mathbf{m}$ is maintained. One can always make τ, π, η small enough such that the obtained sequence of (6) never reaches 0 for the free counter in between. Of course, for the obtained sequence, the h may be smaller (but still positive). The process can be repeated until all the f-increment modes are tried out. The case when $\mathbf{m}$ is an f-decrement mode can be handled similarly. Eventually, an execution path in the form of (6) can be obtained.

So far, we have shown that $\mathbf{X}_0$ reaches $\mathbf{X}_n$ through (6) iff $B(\mathbf{X}_0, \mathbf{X}_n)$ holds. Notice that one can have a finitely number of different B obtained from all the execution paths (6) of $\mathcal{M}$. Hence, the binary reachability of $\mathcal{M}$ (under the three conditions) is mixed linear, by taking a disjunction over all the possible B's. □

Proof of Theorem 6

Proof. Let $\mathcal{M}$ be a dense multicounter machine with k reversal-bounded counters $x_1, \cdots, x_k$ and a free counter x_0, satisfying the three conditions in Lemma 2. We first modify (Cond3) into (Cond3') as follows:

> (Cond3') on any execution sequence in $\mathcal{M}$ from s_{init} to s_{final}, the initial and ending values of the dense free counter are both 0,

by removing the requirement that the dense free counter does not reach 0 in between. Under the conditions of Cond1, Cond2 and Cond3', the result of Lemma 2 still holds, since the execution sequence now can be partitioned into a number (may be unbounded) of phases, in each of which the dense free counter starts and ends with 0 and does not reach 0 in between. In each phase, the starting values of the monotonic counters are not essential. One need only to record the net changes for these counters and sum them up over all the phases. The result then follows from Theorem 2.

In fact, Cond3' can be completely removed. This is because one can construct a machine $\mathcal{M}'$ that works as follows. $\mathcal{M}'$ has monotonic dense counters $z, y_0, y_1, \cdots, y_k$, $x_1, \cdots, x_k$ and a dense free counter x_0. Initially, $\mathcal{M}'$, for each $1 \leq i \leq k$, repeatedly makes fractional increments (0 or more times, nondeterministically guessed) to both y_i and x_i synchronously. Then, $\mathcal{M}'$ repeatedly makes fractional increments to both x_0 and y_0 synchronously. Some moments later, $\mathcal{M}'$ stop the increments and decide to simulate $\mathcal{M}$ from s_{init}. From now on, $\mathcal{M}'$ uses its own counters $x_0, x_1, \cdots, x_k$ to faithfully simulate the same counters in $\mathcal{M}$. During the simulation, all the other counters $z, y_0, y_1, \cdots, y_k$ do not change (always stay). Nondeterministically, when $\mathcal{M}$ reaches s_{final}, $\mathcal{M}'$ stops the simulation and starts the last process. In the process, $\mathcal{M}'$ repeatedly performs fractional decrements to x_0 while, synchronously, makes fractional increments to z. The process ends when x_0 becomes 0. Clearly, at the end, counters $y_0, y_1, \cdots, y_k$ in $\mathcal{M}'$ store the initial value of counters $x_0, x_1, \cdots, x_k$ in $\mathcal{M}$, and counters $z, x_1, \cdots, x_k$ in $\mathcal{M}'$ store the ending values of counters $x_0, x_1, \cdots, x_k$ in $\mathcal{M}$ (assume that z starts from 0 in $\mathcal{M}'$). $\mathcal{M}'$ is still a dense monotonic counter machine with a free counter. Notice that, the dense free counter x_0 in $\mathcal{M}'$ starts and ends with 0. Hence, the binary reachability of $\mathcal{M}'$ is mixed linear. From here, it is easy to conclude so for $\mathcal{M}$.

Now, we are facing with only two conditions: Cond1 and Cond2. Now, we will show that Cond2 can be removed. Suppose that $\mathcal{M}$ satisfies Cond1 but not necessarily Cond2. $\mathcal{M}$ can be simulated by another $\mathcal{M}'$ that satisfies both Cond1 and Cond2. $\mathcal{M}'$ is equipped with many monotonic counters $y_{\mathbf{m}}$ for each mode vector $\mathbf{m}$ of $\mathcal{M}$ in which the component for x_0 is stay. Initially, the y's are 0. $\mathcal{M}'$ starts to faithfully simulate $\mathcal{M}$. However, whenever $\mathcal{M}$ performs a stay move (for the free counter x_0, suppose the mode is $\mathbf{m}$), $\mathcal{M}'$ simulates it with two discrete moves: a discrete increment to x_0 and $y_{\mathbf{m}}$ only, and then a discrete decrement to x_0 only. At the end, $y_{\mathbf{m}}$ stores the number of stay moves of $\mathcal{M}$ with mode $\mathbf{m}$. Notice that in $\mathcal{M}'$, there is no stay moves for the free counter. It is not hard to show that the binary reachability of $\mathcal{M}$ is mixed linear by expressing the binary reachability in terms of the one for $\mathcal{M}'$, whose binary reachability is known mixed linear.

For the only left condition Cond1, one may use the idea in the proof of Theorem 5: a reversal-bounded counters can be simulated by monotonic counters by adding a linear constraint at the end. From here, the result follows. This completes the proof. □