# Playing with Repetitions in Data Words Using Energy Games

Diego Figueira, CNRS, LaBRI, France*
M. Praveen, Chennai Mathematical Institute, India and UMI ReLaX*

**Abstract**

We introduce two-player games which build words over infinite alphabets, and we study the problem of checking the existence of winning strategies. These games are played by two players, who take turns in choosing valuations for variables ranging over an infinite data domain, thus generating multi-attributed *data words*. The winner of the game is specified by formulas in the Logic of Repeating Values, which can reason about repetitions of data values in infinite data words. We prove that it is undecidable to check if one of the players has a winning strategy, even in very restrictive settings. However, we prove that if one of the players is restricted to choose valuations ranging over the Boolean domain, the games are effectively equivalent to *single-sided* games on vector addition systems with states (in which one of the players can change control states but cannot change counter values), known to be decidable and effectively equivalent to energy games.

Previous works have shown that the satisfiability problem for various variants of the logic of repeating values is equivalent to the reachability and coverability problems in vector addition systems. Our results raise this connection to the level of games, augmenting further the associations between logics on data words and counter systems.

## 1 Introduction

Words over an unbounded domain —known as *data words*— is a structure that appears in many scenarios, as abstractions of timed words, runs of counter automata, runs of concurrent programs with an unbounded number or processes, traces of reactive systems, and more broadly as abstractions of any record of the run of processes handling unbounded resources. Here, we understand data word as a (possibly infinite) word in which every position carries a vector of elements from a possibly infinite domain (*e.g.*, a vector of numbers).

Many specification languages have been proposed to specify properties of data words, both in terms of automata [16, 19] and logics [5, 12, 14, 13]. One of the most basic mechanisms for expressing properties on these structures is based on whether a data value at a given position is repeated either *locally* (*e.g.*, in the 2$^{\text{nd}}$ component of the vector at the 4$^{\text{th}}$ future position), or *remotely* (*e.g.*, in the 1$^{\text{st}}$ component of a vector at some position in the past). This has lead to the study of linear temporal logic extended with these kind of tests, called *Logic of Repeating Values* (LRV) [10]. The satisfiability problem for LRV is reducible to and from the reachability problem for Vector Addition Systems with States (VASS), and when the logic is restricted to testing remote repetitions only in the future, it is inter-reducible with the coverability problem for VASS [10, 11]. These connections also extend to data trees and branching VASS [3].

Previous works on data words has been centered around the satisfiability, containment, or model checking problems. Here, we initiate the study of two-player *games* on such structures, motivated by the synthesis of reactive systems (hardware, operating systems, communication protocols). In this setting, a data word can be seen as a trace of an ongoing computation having an interaction with the environment. Some data values are written by the system and some by the environment. Given a property of data words (in our case specified by an LRV formula), we ask whether there exists a program (*i.e.*, a strategy for system player) making the trace to always satisfy the property irrespective of what the environment does. This can be

*The first author is partially supported by ANR project BRAVAS, grant ANR-17-CE40-0028. The second author is partially supported by a grant from the Infosys foundation.

casted as the existence of a winning strategy for a two-player game that we define to this end. The synthesis problem (related the *realizability problem* of Church [8]) on traces has been exhaustively investigated in the case of Boolean variables under LTL/LDL-definable properties, starting from [17]. To the best of our knowledge there have been no works on the more general setup of infinite domains. This work can be seen as a first step towards considering richer structures, this being the case of an infinite set with an equivalence relation.

**Contributions** We show that in general the existence of a winning strategy for the LRV game is undecidable, and that this holds even in restricted settings. Indeed, we show that undecidability holds even when having only past or only future data repetition tests; when working on data words where each player can place only one data value at each position of the data word. These results are shown by reductions from the reachability problem for Minsky machines, simulating counter values by the cardinalities of sets of data values whose repetitions satisfy some properties expressed by LRV formulas.

On the positive side, we show that if the environment player has only Boolean variables instead of data variables ranging over an infinite domain, checking for the existence of a winning strategy becomes decidable. This decidability result is shown via a reduction to the so-called single-sided VASS games [1]. Conversely, we also show that there is a (polytime) reduction from single-sided VASS games to single-sided LRV games. To motivate this restriction on the environment player practically, think of a scenario in which the system is trying to schedule tasks on processors. The number of tasks can be unbounded and task identifiers can be data values. LRV formulas can be used to specify properties such as 'allocated tasks must have been previously initialized'. In the general case where the environment can impose restrictions on which tasks are allocated, our results show undecidability. Suppose the environment can only decide which processors have crashed; this information can be coded using finitely many Boolean variables for the boundedly many processors. In this case our results show decidability.

**Related works** The relations between satisfiability of various logics over data words and the problem of language emptiness for automata models have been explored before. In [5], satisfiability of the two variable fragment of first-order logic on data words is related to reachability in VASS. In [12], satisfiability of LTL extended with freeze quantifiers is related to register automata.

Two player games on counter systems are almost always undecidable and restrictions are imposed to get decidable models. A general framework for games over infinite-state systems with a well-quasi ordering is introduced in [2] and the restriction of downward closure is imposed to get decidability. In [18], the two players follow different of rules, making the abilities of the two players asymmetric and leading to decidability. A possibly infinitely branching version of VASS is studied in [6], where decidability is obtained in the restricted case when the goal of the game is to reach a configuration in which one of the counters has the value zero. Games on VASS with inhibitor arcs are studied in [4] and decidability is obtained in the case where one of the players can only increment counters and the other player can not test for zero value in counters. In [7], energy games are studied, which are games on counter systems and the goal of the game is to play for ever without any counter going below zero in addition to satisfying parity conditions on the control states that are visited infinitely often. Energy games are further studied in [1], where they are related to single-sided VASS games, which restrict one of the players to not make any changes to the counters. Closely related perfect half-space games are studied in [9], where it is shown that optimal complexity upper bounds can be obtained for energy games by using perfect half space games.

**Organization** In Section 2 we define the logic LRV, as well as counter machines and VASS games. In Section 3 we introduce LRV games. Section 4 shows undecidability results for the fragment of LRV with data repetition tests restricted to past only. Section 5 shows the decidabilty result of past-looking single-sided LRV games. Section 6 shows undecidability of future-looking single-sided LRV games, showing that in some sense the decidability result is maximal. We conclude in Section 7. Some of the results in the sections mentioned above only have brief proof ideas. Detailed proofs are in the appendix.

## 2 Preliminaries

We denote by $\mathbb{Z}$ the set of integers and by $\mathbb{N}$ the set of non-negative integers. For any set $S$, we denote by $S^*$ (resp. $S^\omega$) the set of all finite (resp. countably infinite) sequences of elements in $S$. For a sequence $\sigma \in S^*$, we denote its length by $|\sigma|$. We denote by $\mathcal{P}(S)$ (resp. $\mathcal{P}^+(S)$) the set of all subsets (resp. non-empty subsets) of $S$.

**Logic of repeating values** We recall the syntax and semantics of the logic of repeating values from [10, 11]. This logic extends the usual propositional linear temporal logic with the ability to reason about repetitions of data values from an infinite domain. We let this logic use both Boolean variables (*i.e.*, propositions) and variables ranging over an infinite data domain $\mathbb{D}$. The Boolean variables can be simulated by other variables. However, we need to consider fragments of the logic, for which explicitly having Boolean variables is convenient. Let $BVARS = \{q, t, \ldots\}$ be a countably infinite set of Boolean variables ranging over $\{\top, \bot\}$, and let $DVARS = \{x, y, \ldots\}$ be a countably infinite set of 'data' variables ranging over $\mathbb{D}$. We denote by LRV the logic whose formulas are defined as follows:[1]

$$\begin{aligned}\varphi ::= q \mid x \approx \mathsf{X}^j y \mid x \approx \langle \varphi ? \rangle y \mid x \not\approx \langle \varphi ? \rangle y \mid x \approx \langle \varphi ? \rangle^{-1} y \\ \mid x \not\approx \langle \varphi ? \rangle^{-1} y \mid \varphi \wedge \varphi \mid \neg \varphi \mid \mathsf{X}\varphi \mid \varphi \mathsf{U} \varphi \mid \mathsf{X}^{-1}\varphi \\ \mid \varphi \mathsf{S} \varphi \text{ , where } q \in BVARS,\ x, y \in DVARS,\ j \in \mathbb{Z}\end{aligned}$$

A *valuation* is the union of a mapping from $BVARS$ to $\{\top, \bot\}$ and a mapping from $DVARS$ to $\mathbb{D}$. A *model* is a finite or infinite sequence of valuations. We use $\sigma$ to denote models and $\sigma(i)$ denotes the $i^{\text{th}}$ valuation in $\sigma$, where $i \in \mathbb{N} \setminus \{0\}$. For any model $\sigma$ and position $i \in \mathbb{N} \setminus \{0\}$, the satisfaction relation $\models$ is defined inductively as follows. The temporal operators next ($\mathsf{X}$), previous ($\mathsf{X}^{-1}$), until ($\mathsf{U}$) since ($\mathsf{S}$) and its derived operators ($\mathsf{F}$, $\mathsf{G}$, $\mathsf{F}^{-1}$, $\mathsf{G}^{-1}$, etc.) and Boolean connectives are defined in the usual way and are skipped.

$$\begin{aligned}
\sigma, i \models q &\text{ iff } \sigma(i)(q) = \top \\
\sigma, i \models x \approx \mathsf{X}^j y &\text{ iff } 1 \leq i + j \leq |\sigma| \text{ and} \\
&\phantom{\text{ iff }} \sigma(i)(x) = \sigma(i+j)(y) \\
\sigma, i \models x \approx \langle \varphi ? \rangle y &\text{ iff there exists } j > i \text{ such that} \\
&\phantom{\text{ iff }} \sigma(i)(x) = \sigma(j)(y) \text{ and } \sigma, j \models \varphi \\
\sigma, i \models x \not\approx \langle \varphi ? \rangle y &\text{ iff there exists } j > i \text{ such that} \\
&\phantom{\text{ iff }} \sigma(i)(x) \neq \sigma(j)(y) \text{ and } \sigma, j \models \varphi \\
\sigma, i \models x \approx \langle \varphi ? \rangle^{-1} y &\text{ iff there exists } j < i \text{ such that} \\
&\phantom{\text{ iff }} \sigma(i)(x) = \sigma(j)(y) \text{ and } \sigma, j \models \varphi \\
\sigma, i \models x \not\approx \langle \varphi ? \rangle^{-1} y &\text{ iff there exists } j < i \text{ such that} \\
&\phantom{\text{ iff }} \sigma(i)(x) \neq \sigma(j)(y) \text{ and } \sigma, j \models \varphi
\end{aligned}$$

for $q \in BVARS$, $x, y \in DVARS$. Intuitively, the formula $x \approx \mathsf{X}^j y$ tests that the data value mapped to the variable $x$ at the current position repeats in the variable $y$ after $j$ positions. We use the notation $\mathsf{X}^i x \approx \mathsf{X}^j y$ as an abbreviation for the formula $\mathsf{X}^i(x \approx \mathsf{X}^{j-i} y)$ (assuming without any loss of generality that $i \leq j$). The formula $x \approx \langle \varphi ? \rangle y$ tests that the data value mapped to $x$ now repeats in $y$ at a future position that satisfies the nested formula $\varphi$. The formula $x \not\approx \langle \varphi ? \rangle y$ is similar but tests for disequality of data values instead of equality. If a model is being built sequentially step by step and these formulas are to be satisfied at a position, they create obligations (for repeating some data values) to be satisfied in some future step. The formulas $x \approx \langle \varphi ? \rangle^{-1} y$ and $x \not\approx \langle \varphi ? \rangle^{-1} y$ are similar but test for repetitions of data values in past positions.

We consider syntactic restrictions of LRV, indicated by writing symbols in front of LRV. The symbol $\top$ means that in formulas of the form $x \approx \langle \varphi ? \rangle y$, $\varphi$ has to be $\top$ (*i.e.*, nested formulas are not allowed).

[1] In a previous work [11] this logic was denoted by PLRV (LRV + Past).

The symbol $\approx$ means that formulas of the form $x \not\approx \langle\varphi?\rangle y$ or $x \not\approx \langle\varphi?\rangle^{-1} y$ (disequality constraints) are not allowed . The symbol $\rightarrow$ means that formulas of the form $x \approx \langle\varphi?\rangle^{-1} y$ or $x \not\approx \langle\varphi?\rangle^{-1} y$ (past obligations) are not allowed. The symbol $\leftarrow$ means that formulas of the form $x \approx \langle\varphi?\rangle y$ or $x \not\approx \langle\varphi?\rangle y$ (future obligations) are not allowed. For example, LRV$[\top, \approx, \leftarrow]$ denotes the fragment of LRV in which nested formulas, disequality constraints and future obligations are not allowed. For clarity, we replace $\langle\top?\rangle$ with $\Diamond$ in formulas. E.g., we write $x \approx \langle\top?\rangle y$ as simply $x \approx \Diamond y$.

**Parity games on integer vectors** We recall the definition of games on Vector Addition Systems with States (VASS) from [1]. The game is played between two players: system and environment. A VASS game is a tuple $(Q, C, T, \pi)$ where $Q$ is a finite set of states, $C$ is a finite set of counters, $T$ is a finite set of transitions and $\pi : Q \rightarrow \{1, \ldots, p\}$ is a colouring function that assigns a number to each state. The set $Q$ is partitioned into two parts $Q^e$ (states of environment) and $Q^s$ (states of system). A transition in $T$ is a tuple $(q, op, q')$ where $q, q' \in Q$ are the origin and target states and $op$ is an operation of the form $x++$, $x--$ or $nop$, where $x \in C$ is a counter. We say that a transition of a VASS game belongs to environment if its origin belongs to environment; similarly for system. A VASS game is *single-sided* if every environment transition is of the form $(q, nop, q')$. It is assumed that every state has at least one outgoing transition.

A configuration of the VASS game is an element $(q, \vec{n})$ of $Q \times \mathbb{N}^C$, consisting of a state $q$ and a valuation $\vec{n}$ for the counters. A play of the VASS game begins at a designated initial configuration. The player owning the state of the current configuration (say $(q, \vec{n})$) choses an outgoing transition (say $(q, op, q')$) and changes the configuration to $(q', \vec{n'})$ where $\vec{n'}$ is determined by $op$ as follows, where $y$ is any counter not equal to $x$:

- $op = x++$: $\vec{n'}(x) = \vec{n}(x) + 1$, $\vec{n'}(y) = \vec{n}(y)$.
- $op = x--$: $\vec{n'}(x) = \vec{n}(x) - 1$, $\vec{n'}(y) = \vec{n}(y)$.
- $op = nop$: $\vec{n'} = \vec{n}$.

We denote this update as $(q, \vec{n}) \xrightarrow{(q,op,q')} (q', \vec{n'})$. The play is then continued similarly by the owner of the state of the next configuration. If any player wants to take a transition that decrements some counter, that counter should have a non-zero value before the transition. Note that in a single-sided VASS game, environment cannot change the value of the counters. The game continues forever and results in an infinite sequence of configurations $(q_0, \vec{n_0})(q_1, \vec{n_1}) \cdots$. System wins the game if the maximum colour occurring infinitely often in $\pi(q_0)\pi(q_1)\pi(q_2) \cdots$ is even. We assume without loss of generality that from any configuration, at least one transition is enabled (if this condition is not met, we can add extra states and transitions to create an infinite loop ensuring that the owner of the deadlocked configuration looses). In our constructions, we use a generalized form of transitions $q \xrightarrow{\vec{u}} q'$ where $\vec{u} \in \mathbb{Z}^C$, to indicate that each counter $c$ should be updated by adding $\vec{u}(c)$. Such VASS games can be effectively translated into ones of the form defined in the previous paragraph, preserving winning regions.

A strategy $se$ for environment in a VASS game is a mapping $se : (Q \times \mathbb{N}^C)^* \cdot (Q^e \times \mathbb{N}^C) \rightarrow T$ such that for all $\gamma \in (Q \times \mathbb{N}^C)^*$, all $q^e \in Q^e$ and all $\vec{n} \in \mathbb{N}^C$, $se(\gamma \cdot (q^e, \vec{n}))$ is a transition whose source state is $q^e$. A strategy $ss$ for system is a mapping $ss : (Q \times \mathbb{N}^C)^* \cdot (Q^s \times \mathbb{N}^C) \rightarrow T$ satisfying similar conditions. Environment plays a game according to a strategy $se$ if the resulting sequence of configurations $(q_0, \vec{n_0})(q_1, \vec{n_1}) \cdots$ is such that for all $i \in \mathbb{N}$, $q_i \in Q^e$ implies $(q_i, \vec{n_i}) \xrightarrow{se((q_0,\vec{n_0})(q_1,\vec{n_1})\cdots(q_i,\vec{n_i}))} (q_{i+1}, n_{i+1})$. The notion is extended to system player similarly. A strategy $ss$ for system is winning if system wins all the games that she plays according to $ss$, irrespective of the strategy used by environment. It was shown in [1] that it is decidable to check whether system has a winning strategy in a given single-sided VASS game and an initial configuration. An optimal double exponential upper bound was shown for this problem in [9].

**Counter machines** A 2-counter machine is a tuple $(Q, \delta)$, where $Q$ is a finite set of states and $\delta$ is a finite set of transitions. Each transition is a triple of the form $(q_1, u, q_2)$, where $q_1, q_2 \in Q$ and $u$ is either '$c_i--$', '$c_i++$', or '$c_i = 0$?' for some $i \in \{1, 2\}$. The symbols $c_1$, $c_2$ denote counters that the transitions can update. A configuration of the 2-counter machine is a triple $(q, n_1, n_2)$ where $q \in Q$ and $n_1, n_2 \in \mathbb{N}$. The transition relation $\rightarrow$ on configurations is defined as follows. We have $(q_1, n_1, n_2) \rightarrow (q_2, n_1', n_2')$ iff either

- $(q_1, c_i++, q_2) \in \delta$ for $i \in \{1, 2\}$ and $n_i' = n_i + 1$, $n_{3-i}' = n_{3-i}$; or

- $(q_1, c_i --, q_2) \in \delta$ for $i \in \{1,2\}$ and $n_i > 0$, $n'_i = n_i - 1$, $n'_{3-i} = n_{3-i}$; or
- $(q_1, c_i = 0?, q_2) \in \delta$ for $i \in \{1,2\}$ and $n_i = 0$, $(n'_1, n'_2) = (n_1, n_2)$.

Given a 2-counter machine $(Q, \delta)$ and two of its states $q_{init}, q_{fin} \in Q$, the reachability problem is to determine if there is a sequence of transitions of the 2-counter machine starting from the configuration $(q_{init}, 0, 0)$ and ending at the configuration $(q_{fin}, n_1, n_2)$ for some $n_1, n_2 \in \mathbb{N}$. It is known that the reachability problem for 2-counter machines is undecidable [15]. To simplify our undecidability results we further assume, without any loss of generality, that there exists a transition $\hat{t} = (q_{fin}, c_1 ++, q_{fin}) \in \delta$.

# 3 Game of repeating values

The game of repeating values is played between environment and system players, building a sequence of valuations for the variables in $BVARS$ and $DVARS$. The set $BVARS$ is partitioned as $BVARS^e, BVARS^s$, owned by environment and system respectively. The set $DVARS$ is partitioned similarly. Let $B\Upsilon^e$ (resp. $D\Upsilon^e$, $B\Upsilon^s$, $D\Upsilon^s$) be the set of all mappings $bv^e : BVARS^e \to \{\top, \bot\}$ (resp., $dv^e : DVARS^e \to \mathbb{D}$, $bv^s : BVARS^s \to \{\top, \bot\}$, $dv^s : DVARS^s \to \mathbb{D}$). Given two mappings $\upsilon_1 : V_1 \to \mathbb{D} \cup \{\top, \bot\}$, $\upsilon_2 : V_2 \to \mathbb{D} \cup \{\top, \bot\}$ for disjoint sets of variables $V_1, V_2$, we denote by $\upsilon = \upsilon_1 \oplus \upsilon_2$ the mapping defined as $\upsilon(x_1) = \upsilon_1(x_1)$ for all $x_1 \in V_1$ and $\upsilon(x_2) = \upsilon_2(x_2)$ for all $x_2 \in V_2$. Let $\Upsilon^e$ (resp., $\Upsilon^s$) be the set of mappings $\{bv^e \oplus dv^e \mid bv^e \in B\Upsilon^e, dv^e \in D\Upsilon^e\}$ (resp. $\{bv^s \oplus dv^s \mid bv^s \in B\Upsilon^s, dv^s \in D\Upsilon^s\}$). The first round of a game of repeating values is begun by environment choosing a mapping $\upsilon_1^e \in \Upsilon^e$, to which system responds by choosing a mapping $\upsilon_1^s \in \Upsilon^s$. Then environment continues with the next round by choosing a mapping from $\Upsilon^e$ and so on. The game continues forever and results in an infinite model $\sigma = (\upsilon_1^e \oplus \upsilon_1^s)(\upsilon_2^e \oplus \upsilon_2^s) \cdots$. The winning condition is given by a LRV formula $\varphi$ — system wins iff $\sigma, 1 \models \varphi$.

Let $\Upsilon$ be the set of all valuations. For any model $\sigma$ and $i > 0$, let $\sigma \restriction i$ denote the valuation sequence $\sigma(1) \cdots \sigma(i)$, and $\sigma \restriction 0$ denote the empty sequence. A strategy for environment is a mapping $te : \Upsilon^* \to \Upsilon^e$. A strategy for system is a mapping $ts : \Upsilon^* \cdot \Upsilon^e \to \Upsilon^s$. We say that environment plays according to a strategy $te$ if the resulting model $(\upsilon_1^e \oplus \upsilon_1^s)(\upsilon_2^e \oplus \upsilon_2^s) \cdots$ is such that $\upsilon_i^e = te(\sigma \restriction (i-1))$ for all positions $i \in \mathbb{N} \setminus \{0\}$. System plays according to a strategy $ts$ if the resulting model is such that $\upsilon_i^s = ts(\sigma \restriction (i-1) \cdot \upsilon_i^e)$ for all positions $i \in \mathbb{N} \setminus \{0\}$. A strategy $ts$ for system is winning if system wins all games that she plays according to $ts$, irrespective of the strategy used by environment. Given a formula $\varphi$ in (some fragment of) LRV, we are interested in the decidability of checking whether system has a winning strategy in the game of repeating values whose winning condition is $\varphi$.

# 4 Undecidability of LRV[$\top$, $\approx$, $\leftarrow$] games

Here we establish that determining if system has a winning strategy in the LRV[$\top, \approx, \leftarrow$] game is undecidable. This uses a fragment of LRV in which there are no future demands, no disequality demands $\not\approx$, and every sub-formula $x \approx \langle \varphi? \rangle^{-1} y$ is such that $\varphi = \top$. Further, this undecidability result holds even for the case where each player controls only one data variable, and where the distance of local demands is bounded by 3, that is, all local demands of the form $x \approx X^i y$ are so that $-3 \le i \le 3$. Simply put, the result shows that bounding the distance of local demands and the number of data variables does not help in obtaining decidability.

**Theorem 1.** *The winning strategy existence problem for the* LRV[$\top, \approx, \leftarrow$] *game is undecidable, even when each player controls only one variable, and the distance of local demands is bounded by* 3.

As we shall see in the next section, if we further restrict the game so that environment does not control any data variable, we obtain decidability.

Undecidability is shown by reduction from the reachability problem for 2-counter machines. The reduction will be first shown for the case where environment controls a data variable $y$ and system controls a data variable $x$ plus some other Boolean variables, which will encode *labels*. In a second part we will show how to eliminate these Boolean variables.

## 4.1 Reduction with Boolean variables

**Lemma 1.1.** *The winning strategy existence problem for the* LRV$[\top, \approx, \leftarrow]$ *game is undecidable when* environment *controls one data variable and unboundedly many Boolean variables, and* system *one (data) variable.*

*Proof idea.* For convenience, we name the counters of the 2-counter machines $c_x$ and $c_y$ instead of $c_1$ and $c_2$. To simulate counters $c_x$ and $c_y$, we use the environment variable $x$ and system variable $y$. There are a few more Boolean variables that environment uses for the simulation. We define a LRV$[\top, \approx, \leftarrow]$ formula to force environment and system to simulate runs of 2-counter machines as follows. Suppose $\sigma$ is the concrete model built during a game. The value of counter $c_x$ (resp. $c_y$) before the $i^{\text{th}}$ transition is the cardinality of the set $\{d \in \mathbb{D} \mid \exists j \in \{1, \ldots, i\} : \sigma(j)(x) = d, \forall j' \in \{1, \ldots, i\} : \sigma(j')(y) \neq d\}$ (resp. $\{d \in \mathbb{D} \mid \exists j \in \{1, \ldots, i\}, \sigma(j)(y) = d, \forall j' \in \{1, \ldots, i\}, \sigma(j')(x) \neq d\}$). Intuitively, the value of counter $c_x$ is the number of data values that have appeared under variable $x$ but not under $y$. In each round, environment chooses the transition of the 2-counter machine to be simulated and sets values for its variables accordingly. If everything is in order, system cooperates and sets the value of the variable $y$ to complete the simulation. Otherwise, system can win immediately by setting the value of $y$ to a value that certifies that the actions of environment violate the semantics of the 2-counter machine. If any player deviates from this behaviour at any step, the other player wins immediately. The only other way system can win is by reaching the halting state and the only other way environment can win is by properly simulating the 2-counter machine for ever and never reaching the halting state. $\square$

## 4.2 Getting rid of Boolean variables

The reduction above makes use of some Boolean variables to enode transitions of the 2-counter machine. However, one can modify the reduction above to do the encoding inside equivalence classes of the variable $x$. Suppose there are $m-1$ labels that we want to encode. A data word prefix of the form

$$\begin{array}{rcccc} \textit{label}: & l_1 & l_2 & & l_n \\ x: & x_1 & x_2 & \cdots & x_n \\ y: & y_1 & y_2 & & y_n \end{array}$$

where $l_i$ is the label of position $i$, $x_i$ is the data value of $x$ in position $i$, and $y_i$ is the data value of $y$ in position $i$, will be now encoded as

$$\begin{array}{rcccccccccccccccc} x: & d & d & \multirow{2}{*}{$w_1$} & x_1 & d & d & \multirow{2}{*}{$w_2$} & x_2 & d & d & \cdots & d & d & \multirow{2}{*}{$w_n$} & x_n & d & d \\ y: & d & d & & y_1 & d & d & & y_2 & d & d & & d & d & & y_n & d & d \end{array}$$

where each $w_i$ is a data word of the form $(d_1, d_1) \cdots (d_m, d_m)$; further the data values of $w_i$ are so that $d \notin \{d_1, \ldots, d_m\}$, and so that every pair of $w_i, w_j$ with $i \neq j$ has disjoint sets of data values. The purpose of $w_i$ is to encode the label $l_i$; the purpose of the repeated data value $(d, d)$ is to delimit the boundaries of each encoding of a label, which we will call a 'block'; the purpose of repeating $(d, d)$ at each occurrence is to avoid confusing this position with the encoding position $(x_i, y_i)$ —*i.e.*, a boundary position is one whose data value is repeated at distance m+3 *and* at distance 1.

This encoding can be enforced using a LRV formula, and in particular one can test with a formula $\varphi_{block(i)}$ whether we are standing in the $i$-th position of a block. Inside any $w_j$, the $i$-th label is encoded as the first position of $w_j$ being equal to the $i$-th position of $w_j$. Finally, all the formulas used in the previous reduction can be adapted so that they express properties of the $(x_i, y_i)$ part of the block. Each time a property said "there exists a position where $\psi$ holds", now it should say "there exists a position where $\varphi_{block(m+2)} \wedge \psi$ holds", where it isaid "for every position $\psi$ holds" it should now say "for every position $\varphi_{block(m+2)} \Rightarrow \psi$ holds". Finally, and most importantly, the encoding of values of counters in the reduction is not broken since the additional positions have the property of having the same data value under $x$ as under $y$, and in this the encoding of counter $c_x$ —*i.e.*, the numbrer of data values that have appeared under $x$ but not under $y$— is not modified; similarly for counter $c_y$.

**Lemma 1.2.** *The winning strategy existence problem for the* LRV$[\top, \approx, \leftarrow]$ *game is undecidable when* system *and* environment *controls one (data) variable each and no Boolean variables.*

### 4.3 Getting rid of unbounded local tests

The previous two undecidability results show that determining if a player has a winning strategy on the LRV game is undecidable as soon as we have an unbounded number of variables or a bounded number of variables but an unbounded $\mathsf{X}$-distance of local demands. One question that emerges from these results is whether these undecidability results extend also to the setting with boundedly many variables as well as bounded $\mathsf{X}$-distance of local demands. However, in the last reduction, through a more clever encoding one can avoid testing whether two positions at distance $n$ have the same data value by a chained series of tests. This is a standard coding which does not break the 2-counter machine reduction, assuming that, as before, all new positions added share the same data value under $x$ as under $y$.

Then we obtain the following, which proves the theorem.

**Lemma 1.3.** *The winning strategy existence problem for the* LRV$[\top, \approx, \leftarrow]$ *game is undecidable when* system *and* environment *control only one variable each, and the distance of local data repetition demands is bounded by 3.*

## 5 Single-sided LRV$[\top, \leftarrow]$ and VASS games

In this section we show that the LRV$[\leftarrow, \top]$-game is decidable. We first observe that we do not need to consider $\not\approx$ formulas for our decidability argument, since there is a reduction of the existence of winning strategy problem that removes all sub-formulas of the from $x \not\approx \Diamond^{-1} y$.

**Proposition 1.1.** *There is a polynomial-time reduction from the existence of winning strategy problem for* LRV$[\leftarrow, \top]$ *into the problem on* LRV$[\leftarrow, \top, \approx]$.

This is done in a similar way as it was done for the satisfiability problem [11, Proposition 4]. The key observation is that

- $\neg(x \not\approx \Diamond^{-1} y)$ is equivalent to $\neg \mathsf{X}^{-1}\top \vee (x \approx \mathsf{X}^{-1} y \wedge \mathsf{G}^{-1}(\neg \mathsf{X}^{-1}\top \vee y \approx \mathsf{X}^{-1} y))$;
- $x \not\approx \Diamond^{-1} y$ can be translated into $\neg(x \approx x_{\approx\Diamond^{-1}y}) \wedge x_{\approx\Diamond^{-1}y} \approx \Diamond^{-1} y$ for a new variable $x_{\approx\Diamond^{-1}y}$ belonging to the same player as $x$.

Given a formula $\varphi$ in negation normal form, consider the formula $\varphi'$ resulting from the replacements listed above. It follows that $\varphi'$ does not make use of $\not\approx$. It is easy to see that there is a winning strategy for system in the game with winning condition $\varphi$ if and only if she has a winning strategy for the game with condition $\varphi'$.

We consider games in which the environment player only owns Boolean variables while the system player owns variables ranging over $\mathbb{D}$, which can also be used to simulate Boolean variables. We call this the single-sided LRV$[\top, \leftarrow]$ games and show that checking whether the system player has a winning strategy is decidable. The main concept we use for decidability is a symbolic representation of models, introduced in [10]. The building blocks of the symbolic representation are *frames*, which we adapt here. We finally show effective reductions between single-sided LRV$[\top, \leftarrow]$ games and single-sided VASS games. This implies decidability of single-sided LRV$[\top, \leftarrow]$ games. From Proposition 1.1, it suffices to show effective reductions between single-sided LRV$[\top, \approx, \leftarrow]$ games and single-sided VASS games.

Given a formula in LRV$[\top, \approx, \leftarrow]$, we replace sub-formulas of the form $x \approx \mathsf{X}^{-j} y$ with $\mathsf{X}^{-j}(y \approx \mathsf{X}^{j} x)$ if $j > 0$. For a formula $\varphi$ obtained after such replacements, let $l$ be the maximum $i$ such that a term of the form $\mathsf{X}^i x$ appears in $\varphi$. We call $l$ the $\mathsf{X}$-length of $\varphi$. Let $BVARS^{\varphi} \subseteq BVARS$ and $DVARS^{\varphi} \subseteq DVARS$ be the set of Boolean and data variables used in $\varphi$. Let $\Omega_l^{\varphi}$ be the set of constraints of the form $\mathsf{X}^i q$, $\mathsf{X}^i x \approx \mathsf{X}^j y$ or $\mathsf{X}^i(x \approx \Diamond^{-1} y)$, where $q \in BVARS^{\varphi}$, $x, y \in DVARS^{\varphi}$ and $i, j \in \{0, \ldots, l\}$. For $e \in \{0, \ldots, l\}$, an $(e, \varphi)$-frame is a set of constraints $fr \subseteq \Omega_l^{\varphi}$ that satisfies the following conditions:

(F0) For all constraints $\mathsf{X}^i q, \mathsf{X}^i x \approx \mathsf{X}^j y, \mathsf{X}^i(x \approx \Diamond^{-1} y) \in fr$, $i, j \in \{0, \ldots, e\}$.

(F1) For all $i \in \{0, \ldots, e\}$ and $x \in DVARS^{\varphi}$, $\mathsf{X}^i x \approx \mathsf{X}^i x \in fr$.

(F2) For all $i, j \in \{0, \ldots, e\}$ and $x, y \in DVARS^{\varphi}$, $\mathsf{X}^i x \approx \mathsf{X}^j y \in fr$ iff $\mathsf{X}^j y \approx \mathsf{X}^i x \in fr$.

(F3) For all $i, j, j' \in \{0, \ldots, e\}$ and $x, y, z \in DVARS^{\varphi}$, if $\{\mathsf{X}^i x \approx \mathsf{X}^j y, \mathsf{X}^j y \approx \mathsf{X}^{j'} z\} \subseteq fr$, then $\mathsf{X}^i x \approx \mathsf{X}^{j'} z \in fr$.

(F4) For all $i, j \in \{0, \ldots, e\}$ and $x, y \in DVARS^{\varphi}$ such that $\mathsf{X}^i x \approx \mathsf{X}^j y \in fr$:

- if $i = j$, then for every $z \in DVARS^{\varphi}$ we have $\mathsf{X}^i (x \approx \Diamond^{-1} z) \in fr$ iff $\mathsf{X}^j (y \approx \Diamond^{-1} z) \in fr$.
- if $i < j$, then $\mathsf{X}^j (y \approx \Diamond^{-1} x) \in fr$ and for any $z \in DVARS^{\varphi}$, $\mathsf{X}^j (y \approx \Diamond^{-1} z) \in fr$ iff either $\mathsf{X}^i (x \approx \Diamond^{-1} z) \in fr$ or there exists $i \leq j' < j$ with $\mathsf{X}^j y \approx \mathsf{X}^{j'} z \in fr$.

The condition (F0) ensures that a frame can constrain at most $(e+1)$ contiguous valuations. The next three conditions ensure that equality constraints in a frame form an equivalence relation. The last condition ensures that obligations for repeating values in the past are consistent among various variables.

A pair of $(l, \varphi)$-frames $(fr, fr')$ is said to be one-step consistent iff

(O1) for all $\mathsf{X}^i x \approx \mathsf{X}^j y \in \Omega_l^{\varphi}$ with $i, j > 0$, we have $\mathsf{X}^i x \approx \mathsf{X}^j y \in fr$ iff $\mathsf{X}^{i-1} x \approx \mathsf{X}^{j-1} y \in fr'$,

(O2) for all $\mathsf{X}^i (x \approx \Diamond^{-1} y) \in \Omega_l^{\varphi}$ with $i > 0$, we have $\mathsf{X}^i (x \approx \Diamond^{-1} y) \in fr$ iff $\mathsf{X}^{i-1} (x \approx \Diamond^{-1} y) \in fr'$ and

(O3) for all $\mathsf{X}^i q \in \Omega_l^{\varphi}$ with $i > 0$, we have $\mathsf{X}^i q \in fr$ iff $\mathsf{X}^{i-1} q \in fr'$.

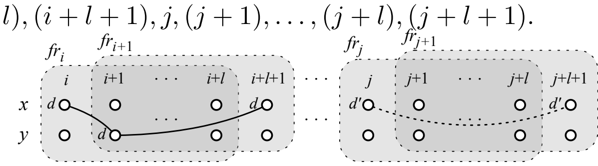
For $e \in \{0, \ldots, l-1\}$, an $(e, \varphi)$ frame $fr$ and an $(e+1, \varphi)$ frame $fr'$, the pair $(fr, fr')$ is said to be one step consistent iff $fr \subseteq fr'$ and for every constraint in $fr'$ of the form $\mathsf{X}^i x \approx \mathsf{X}^j y$, $\mathsf{X}^i q$ or $\mathsf{X}^i (x \approx \Diamond^{-1} y)$ with $i, j \in \{0, \ldots, e\}$, the same constraint also belongs to $fr$.

An (infinite) $(l, \varphi)$-symbolic model $\rho$ is an infinite sequence of $(l, \varphi)$-frames such that for all $i \in \mathbb{N}$, the pair $(\rho(i), \rho(i+1))$ is one-step consistent. Let us define the symbolic satisfaction relation $\rho, i \models_{symb} \varphi'$ where $\varphi'$ is a sub-formula of $\varphi$. The relation $\models_{symb}$ is defined in the same way as $\models$ for LRV, except that for every element $\varphi'$ of $\Omega_l^{\varphi}$, we have $\rho, i \models_{symb} \varphi'$ whenever $\varphi' \in \rho(i)$. We say that a concrete model $\sigma$ realizes a symbolic model $\rho$ if for every $i \in \mathbb{N} \setminus \{0\}$, $\rho(i) = \{\varphi' \in \Omega_l^{\varphi} \mid \sigma, i \models \varphi'\}$. The following result follows easily from definitions.

**Lemma 1.4** (symbolic vs. concrete models)**.** *Suppose $\varphi$ is a* LRV$[\top, \approx, \leftarrow]$ *formula of* $\mathsf{X}$*-length $l$, $\rho$ is a $(l, \varphi)$-symbolic model and $\sigma$ is a concrete model realizing $\rho$. Then $\rho$ symbolically satisfies $\varphi$ iff $\sigma$ satisfies $\varphi$.*

We fix a LRV$[\top, \approx, \leftarrow]$ formula $\varphi$ of $\mathsf{X}$-length $l$. For $e \in \{0, \ldots, l\}$, an $(e, \varphi)$-frame $fr$, $i \in \{0, \ldots, e\}$ and a variable $x$, the set of past obligations of the variable $x$ at level $i$ in $fr$ is defined to be the set $\mathsf{PO}_{fr}(x, i) = \{y \in DVARS^{\varphi} \mid \mathsf{X}^i (x \approx \Diamond^{-1} y) \in fr\}$. The equivalence class of $x$ at level $i$ in $fr$ is defined to be $[(x, i)]_{fr} = \{y \in DVARS^{\varphi} \mid \mathsf{X}^i x \approx \mathsf{X}^i y \in fr\}$.

Consider a concrete model $\sigma$ restricted to two variables $x, y$ as shown below. The top row indicates the positions $i, (i+1), \ldots, (i+l), (i+l+1), j, (j+1), \ldots, (j+l), (j+l+1)$.




The left column indicates the two variables $x, y$ and the remaining columns indicate valuations. E.g., $\sigma(i+1)(y) = d$ and $\sigma(j+l+1)(x) = d'$. Let $fr_i = \{\varphi' \in \Omega_l^{\varphi} \mid \sigma, i \models \varphi'\}$. We have indicated this pictorially by highlighting the valuations that determine the contents of $fr_i$. The data values for $x$ at positions $i$ and $(i+l+1)$ are equal, but the positions are too far apart to be caputured by any one constraint of the form $\mathsf{X}^{\alpha} x \approx \mathsf{X}^{\beta} x$ in $\Omega_l^{\varphi}$. However, the intermediate position $(i+1)$ has the same data value and is less than $l$ positions apart from both positions. One constraint from $\Omega_l^{\varphi}$ can capture the data repetition between positions $i$ and $(i+1)$ while another one captures the repetition between positions $(i+1)$ and $(i+l+1)$, thus indirectly capturing the repetition between positions $i$ and $(i+l+1)$. For $e \in \{0, \ldots, l\}$, an $(e, \varphi)$-frame $fr$, $i \in \{0, \ldots, e\}$ and a variable $x$, we say that there is a forward (resp. backward) reference from $(x, i)$ in $fr$ if $\mathsf{X}^i x \approx \mathsf{X}^{i+j} y \in fr$ (resp. $\mathsf{X}^i x \approx \mathsf{X}^{i-j} y \in fr$) for some $j > 0$ and $y \in DVARS^{\varphi}$. The constraint $x \approx \mathsf{X} y$ in $fr_i$ above is a forward reference from $(x, 0)$ in $fr_i$, while the constraint $\mathsf{X}^l x \approx y$ is a backward reference from $(x, l)$ in $fr_{i+1}$.

In the above picture, the data values of $x$ at positions $j$ and $(j+l+1)$ are equal, but the two positions are too far apart to be captured by any constraint of the form $\mathsf{X}^{\alpha} z \approx \mathsf{X}^{\beta} w$ in $\Omega_l^{\varphi}$. Neither are there any intermediate positions with the same data value to capture the repetition indirectly. We maintain a counter to keep track of the number of such remote data repetitions. Let $X \subseteq DVARS^{\varphi}$ be a set of variables. A *point of decrement* for counter $X$ in an $(e,\varphi)$-frame $fr$ is an equivalence class of the form $[(x,e)]_{fr}$ such that there is no backward reference from $(x,e)$ in $fr$ and $\mathsf{PO}_{fr}(x,e) = X$. In the above picture, the equivalence class $[(x,l)]_{fr_{j+1}}$ in the frame $fr_{j+1}$ is a point of decrement for $\{x\}$. A *point of increment for $X$ in an $(l,\varphi)$-frame $fr$* is an equivalence class of the form $[(x,0)]_{fr}$ such that there is no forward reference from $(x,0)$ in $fr$ and $[(x,0)]_{fr} \cup \mathsf{PO}_{fr}(x,0) = X$. In the above picture, the equivalence class $[(x,0)]_{fr_j}$ in the frame $fr_j$ is a point of increment for $\{x\}$. Points of increment are not present in $(e,\varphi)$-frames for $e < l$ since such frames do not contain complete information about constraints in the next $l$ positions. We denote by $inc(fr)$ the vector indexed by non-empty subsets of $DVARS^{\varphi}$, where each coordinate contains the number of points of increment in $fr$ for the corresponding subset of variables. Similarly, we have the vector $dec(fr)$ for points of decrement.

Given a LRV[$\top$, $\approx$, $\leftarrow$] formula $\varphi$ with $DVARS^{e} = BVARS^{s} = \emptyset$, we construct a single-sided VASS game as follows. Let $l$ be the $\mathsf{X}$-length of $\varphi$ and $\mathsf{FR}$ be the set of all $(e,\varphi)$-frames for all $e \in \{0,\ldots,l\}$. Let $A^{\varphi}$ be a parity automaton that accepts a symbolic model iff it symbolically satisfies $\varphi$, with set of states $Q^{\varphi}$ and initial state $q^{\varphi}_{init}$. The single-sided VASS game will have set of counters $\mathcal{P}^{+}(DVARS^{\varphi})$, set of environment states $\{-1,0,\ldots,l\} \times Q^{\varphi} \times (\mathsf{FR} \cup \{\bot\})$ and set of system states $\{-1,0,\ldots,l\} \times Q^{\varphi} \times (\mathsf{FR} \cup \{\bot\}) \times \mathcal{P}(BVARS^{\varphi})$. Every state will inherit the colour of its $Q^{\varphi}$ component. The intention of this single-sided VASS game is to make the two players build a sequence of frames. For convenience, we let $\bot$ to be the only $(-1,\varphi)$-frame and $(\bot, fr')$ be one-step consistent for every 0-frame $fr'$. The initial state is $(-1, q^{\varphi}_{init}, \bot)$, the initial counter values are all 0 and the transitions are as follows, where we have used $\lceil \cdot \rceil l$ to denote the mapping that is identity on $\{-1,0,\ldots,l-1\}$ and maps all others to $l$.

- $(e,q,fr) \xrightarrow{\vec{0}} (e,q,fr,V)$ for every $e \in \{-1,0,\ldots,l\}$, $q \in Q^{\varphi}$, $fr \in \mathsf{FR} \cup \{\bot\}$ and $V \subseteq BVARS^{\varphi}$.
- $(e,q^{\varphi}_{init},fr,V) \xrightarrow{inc(fr)-dec(fr')} (e+1,q^{\varphi}_{init},fr')$ for every $V \subseteq BVARS^{\varphi}$, $e \in \{-1,0,\ldots,l-2\}$, $(e,\varphi)$-frame $fr$ and $(e+1,\varphi)$-frame $fr'$, where the pair $(fr,fr')$ is one-step consistent and $\{p \in BVARS^{\varphi} \mid \mathsf{X}^{e+1}p \in fr'\} = V$.
- $(e,q,fr,V) \xrightarrow{inc(fr)-dec(fr')} (\lceil e+1 \rceil l, q', fr')$ for every $e \in \{l-1,l\}$, $(e,\varphi)$-frame $fr$, $V \subseteq BVARS^{\varphi}$, $q,q' \in Q^{\varphi}$ and $(\lceil e+1 \rceil l,\varphi)$-frame $fr'$, where the pair $(fr,fr')$ is one-step consistent, $\{p \in BVARS^{\varphi} \mid \mathsf{X}^{\lceil e+1 \rceil l}p \in fr'\} = V$ and $q \xrightarrow{fr'} q'$ is a transition in $A^{\varphi}$.

Transitions of the form $(e,q,fr) \xrightarrow{\vec{0}} (e,q,fr,V)$ let the environment choose any subset $V$ of $BVARS^{\varphi}$ to be true in the next round. In transitions of the form $(e,q,fr,V) \xrightarrow{inc(fr)-dec(fr')} (\lceil e+1 \rceil l, q', fr')$, the condition $\{p \in BVARS^{\varphi} \mid \mathsf{X}^{\lceil e+1 \rceil l}p \in fr'\} = V$ ensures that the frame $fr'$ chosen by the system is compatible with the subset $V$ of $BVARS^{\varphi}$ chosen by the environment in the preceding step. By insisting that the pair $(fr,fr')$ is one-step consistent, we ensure that the sequence of frames built during a game is a symbolic model. The condition $q \xrightarrow{fr'} q'$ ensures that the symbolic model is accepted by $A^{\varphi}$ and hence symbolically satisfies $\varphi$. The update vector $inc(fr) - dec(fr')$ ensures that symbolic models are realizable, as explained in the proof of the following result.

**Lemma 1.5** (repeating values to VASS)**.** *Suppose $\varphi$ is a* LRV[$\top$, $\approx$, $\leftarrow$] *formula with $DVARS^{e} = BVARS^{s} = \emptyset$. Then* system *has a winning strategy in the corresponding single-sided* LRV[$\top$, $\approx$, $\leftarrow$] *game iff she has a winning strategy in the single-sided VASS game constructed above.*

*Proof idea.* A game on the single-sided VASS game results in a sequence of frames. The single-sided VASS game embeds automata which check that these sequences are symbolic models that symbolically satisfy $\varphi$. This in conjunction with Lemma 1.4 (symbolic vs. concrete models) will prove the result, provided the symbolic models are also realizable. Some symbolic models are not realizable since frames contain too many constraints about data values repeating in the past and no concrete model can satisfy all those constraints. To avoid this, the single-sided VASS game maintains counters for keeping track of the number of such constraints. Whenever a frame contains such a past repetition constraint that is not satisfied locally within the frame itself, there is an absence of backward references in the frame and it results in a point of decrement. Then the $-dec(fr')$ part of transitions of the form $(e,q,fr,V) \xrightarrow{inc(fr)-dec(fr')} (\lceil e+1 \rceil l, q', fr')$ will decrement the corresponding counter. In order for this counter to have a value of at least 0, the counter should have been incremented earlier by $inc(fr)$ part of earlier transitions. This ensures that symbolic models resulting from the single-sided VASS games are realizable. ☐

Given a LRV[$\top$, $\approx$, $\leftarrow$] formula with $DVARS^{e} = BVARS^{s} = \emptyset$ and no past-time temporal modalities, the single-sided VASS game defined above is exponential in the size of the formula and can be constructed in exponential time. Hence, the double exponential time upper bound for energy games (and hence for single-sided VASS games) given in [9] translates to triple exponential time for single-sided LRV[$\top$, $\approx$, $\leftarrow$] games.

Next we show that single-sided VASS games can be effectively reduced to single-sided LRV[$\top$, $\approx$, $\leftarrow$] games.

**Theorem 2.** *Given a single-sided VASS game, a single-sided* LRV[$\top$, $\approx$, $\leftarrow$] *game can be constructed in polynomial time such that the system player has a winning strategy in the first game iff the system player has a winning strategy in the second one.*

*Proof idea.* We will simulate runs of single-sided VASS games with models of formulas in LRV. The formulas satisfied at position $i$ of the concrete model will contain information about counter values before the $i^{\text{th}}$ transition and the identity of the $i^{\text{th}}$ transition chosen by the environment and the system players in the run of the single-sided VASS game. For simulating a counter $x$, we use two system variables $x$ and $\overline{x}$. The data values assigned to these variables from positions 1 to $i$ in a concrete model $\sigma$ will represent the counter value that is equal to the cardinality of the set $\{d \in \mathbb{D} \mid \exists j \in \{1, \ldots, i\}, \sigma(j)(x) = d, \forall j' \in \{1, \ldots, i\}, \sigma(j')(\overline{x}) \neq d\}$. Using formulas in LRV[$\top$, $\approx$, $\leftarrow$], the two players of can be enforced to correctly update the concrete model to faithfully reflect the moves in the single-sided VASS game. A formula can also be written to ensure that system wins the single-sided LRV[$\top$, $\approx$,$\leftarrow$] game iff the single-sided VASS game being simulated satisfies the parity condition. ☐

# 6 Single-sided LRV[$\top$, $\approx$, $\rightarrow$] undecidable

In this section we show that the positive decidability result for the single-sided LRV[$\top$,$\leftarrow$] game cannot be replicated for the future demands fragment, even in a restricted setting.

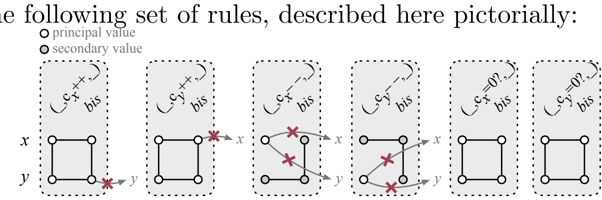
**Theorem 3.** *The existence of winning strategy for the single-sided* LRV[$\top$,$\approx$,$\rightarrow$] *game is undecidable, even when* environment *has only one Boolean variable and* system *has three data variables.*

We don't know the decidability status for the case where system has less than three data variables.

As in the previous undecidability results in Section 4, the result is proven by a reduction from the reachability problem for 2-counter machines. System makes use of *labels* to encode the sequence of transitions of a witnessing run of the counter machine. This time, system has 3 data variables $x, y, z$ (in addition to a number of Boolean variables which encode the labels); and environment has just one Boolean variable $b$. Variables $x$, $y$ are used to encode the counters $c_x$ and $c_y$ as before, and variables $z$, $b$ are used to ensure that there are no 'illegal' transitions — namely, no decrements of a zero-valued counter, and no tests for zero for a non-zero-valued counter.

Each transition in the run of the 2-counter machine will be encoded using *two* consecutive positions of the game. Concretely, while in the previous coding of Section 4 a witnessing reachability run $t_1\, t_2 \cdots t_n \in \delta^*$ was encoded with the label sequence $\mathit{begin}\, t_1\, t_2 \cdots t_n \hat{t}^{\omega}$, in this encoding transitions are interspersed with a special $\mathit{bis}$ label, and thus the run is encoded as $t_1\, \mathit{bis}\, t_2\, \mathit{bis} \cdots t_n\, \mathit{bis}\, (\hat{t}\, \mathit{bis})^{\omega} \in (\delta \cup \{\mathit{bis}\})^{\omega}$.

Suppose a position has the label of a $c_x{+}{+}$ transition and the variable $x$ has the data value $d$. Our encoding will ensure that if the data value $d$ repeats in the future, it will be only once and at a position that has the label of a $c_x{-}{-}$ transition. A symmetrical property holds for $c_y$ and variable $y$. The value of counter $c_x$ (resp. $c_y$) before the $i^{\text{th}}$ transition (encoded in the $2i^{th}$ and $(2i+1)^{st}$ positions) is the number of positions $j < 2i$ satisfying the following two conditions: i) the position $j$ should have the label of a $c_x{+}{+}$ transition and ii) $\sigma(j)(x) \notin \{\sigma(j')(x) \mid j+1 < j' < 2i\}$. Intuitively, if $2i$ is the current position, the value of $c_x$ (resp. $c_y$) is the number of previous positions that have the label of a $c_x{+}{+}$ transition whose data value is not yet matched by a position with the label of a $c_x{-}{-}$ transition. In this reduction we assume that system plays first and environment plays next at each round, since it is easier to understand (the reduction also holds for the game where turns are inverted by shifting environment behaviour by one position; see appendix C for details). At each round, system will play a label *bis* if the last label played was a transition. Otherwise, she will choose the next transition of the 2-counter machine to simulate and she will chose the values for variables $x$, $y$, $z$ in such a way that the aforementioned encoding for counters $c_x$ and $c_y$ is preserved. To this end, system is bound by the following set of rules, described here pictorially:




The first (leftmost) rule, for example, reads that whenever there is a $c_x++$ transition label, then all four values for $x$ and $y$ in both positions (*i.e.*, the transition position and the next *bis* position) must have the same data value $d$ (which we call 'principal'), which does not occur in the future under variable $y$. The third rule says that $c_x--$ is encoded by having $x$ on the first position to carry the 'principal' data value $d$ of the transition, which is *final* (that is, it is not repeated in the future under $x$ or $y$), and all three remaining positions have the same data value $d'$ different from $d$. In this way, system can make sure that the value of $c_x$ is decremented, by playing a data value $d$ that has occurred in a $c_x++$ position that is not yet mathed. (System could also play some other data value which does not match any previous $c_x++$ position, but this 'illegal' move will lead to a losing play for system, as explained later.) In this rule, the usage of two positions per transition becomes essential: it ensures that the data value $d'$ of $y$ (for which $d' \neq d$) appears in the future both in $x$ and $y$. Thus, the presence of $d'$ doesn't affect the value of $c_y$ or $c_x$ —to affect either, the data value should repeat in only one variable. For details on writing these rules in LRV[$\top$, $\approx$, $\rightarrow$], see Section C.

From these rules, it follows that every $c_k++$ can be matched to at most one future $c_k--$. However, there can be two ways in which this coding can fail: a) there could be invalid tests $c_k = 0?$, that is, a situation in which the preceding positions of the test contain a $c_k++$ transition which is not matched with a $c_k--$ transition; and b) there could be some $c_k--$ with no previous matching $c_k++$. As we will see next, variables $z$ and $b$ play a crucial role in the game whenever any of these two cases occurs. In all the rounds, environment always plays $\top$, except if he detects that one of these two situations, a) or b), have arisen, in which case he plays $\perp$. In the following rounds system plays a value in $z$ that will enable to test, with an LRV formula, if there was indeed an a) or b) situation, in which case system will lose, or if environment was just 'bluffing', in which case system will win. Since this is the most delicate point in the reduction, we dedicate the remaining of this section to the explanation of how these two situations a) and b) are treated. (More details in Appendix C.)

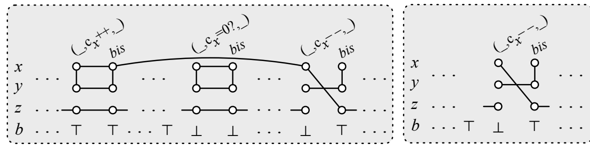
Remember that environment has just *one bit* of information to play with. Further, the LRV property we will build ensures that the sequence of $b$-values must be from the set $\top^*\perp^*\top^\omega$.

**a) Avoiding illegal tests for zero.** Suppose that at some point of the 2-counter machine simulation, system decides to play a $c_k = 0?$ transition. Suppose there is some preceding $c_k++$ transition for which either: a1) there is no matching $c_k--$ transition; or a2) there is a matching $c_k--$ transition but it occurs after the $c_k = 0?$ transition. Situation a1 can be easily avoided by ensuring that any winning play must satisfy the formula

$$\mu = \mathsf{G}(\tau_{(c_k++)} \wedge \mathsf{F}\tau_{(c_k=0?)} \Rightarrow k \approx \Diamond k)$$

for every $k \in \{x, y\}$. Here, $\tau_{inst}$ tests if the current position is labelled with an instruction of type *inst*. On the other hand, Situation a2 requires environment to play a certain strategy (represented in Figure 1-a2). This means that $c_k$ is non-zero at the position of the $c_k = 0?$ transition, and that this is an illegal transition; thus, environment must respond accordingly. Further, suppose this is the *first* illegal transition that has occurred so far. Environment, who so far has been playing only $\top$, decides to play $\perp$ to mark the cheating point. Further, he will continue playing $\perp$ until the matching $c_k--$ transition is reached (if it is never reached, it is situation a1 and system loses as explained before), after which he will play $\top$ forever afterwards. In some sense, environment provides a *link* between the illegal transition and the proof of its illegality through a $\perp^*$-path. The following characterizes environment's denouncement of an illegal test for zero:

*Property 1:* $b$ becomes $\perp$ at a $c_k = 0?$ position and stops being $\perp$ at a $c_k--$ position thereafter.



*situation a2)* *situation b)*

Figure 1: Depiction of best strategies in both situations.

Note that Property 1 is clearly definable by a formula $\pi_1$ of LRV$[\top, \approx, \to]$. If Property 1 holds, a formula $\varphi_1$ can constrain system to play $z$ according to the following: $z$ always carries the same data value, distinct from the values of all other variables, but as soon as the last $\perp$ value is played, which has to be on a $c_k --$ position, the value of $z$ changes and holds the principal value of that $c_k --$ transition,[2] and it continues to hold that value forever after (*cf.* Figure 1-$a2$). Further, if environment cheated in his denouncement by linking a $c_k = 0?$ transition with a future $c_k --$ with a matching $c_k ++$ that falls in-between the test for zero and the decrement, then a property $\pi'_1$ can catch this: there exists a $c_k ++$ with $\perp$ whose principal value matches that of a future $z$-value.

Finally, assuming environment correctly denounced an illegal test for zero and system played accordingly on variable $z$, a property $\varphi'_1$ can test that environment exposed an illegal transition, by testing that there exists a $c_k ++$ transition whose principal value corresponds to the $z$-value of some future position. Thus, the encoding for this situation is expressed with the formula $\psi_1 = \mu \wedge ((\pi_1 \wedge \neg\pi'_1) \Rightarrow (\varphi_1 \wedge \neg\varphi'_1))$.

**b) Avoiding illegal decrements.** Suppose now that at some point of the 2-counter machine simulation, system decides to play a $c_k --$ transition for which there is no preceding $c_k ++$ transition matching its final data value. This is a form of cheating, and thus environment should respond accordingly. Further, suppose this is the first cheating that has occurred so far. Environment, who so far has been playing only $\top$, decides then to mark this position with $\perp$; and for the remaining of the play environment plays only $\top$ (even if more illegal transitions are performed in the sequel). Summing up, for this situation environment's best strategy has a value sequence from $\top^*\perp\top^\omega$, and this property characterizes environment's denouncement of an illegal decrement (*cf.* Figure 1-$b$).

*Property 2:* $b$ becomes $\perp$ at a $c_k --$ position and stops being $\perp$ immediately after.

A formula $\pi_2$ can test Property 2; and a formula $\varphi_2$ can constrain variable $z$ to always have the same data value —distinct from all other data values played on variables $x, y$— while $b$ contains $\top$ values; and as soon as $b$ turns to $\perp$ on a $c_k --$ position, then $z$ at the next position takes the value of the current variable $k$, and maintains that value (*cf.* Figure 1-$b$). Further, a formula $\varphi'_2$ tests that in this case there must be some $c_k ++$ position with a data value equal to variable $z$ of a future position. The final formula for this case is then $\psi_2 = \pi_2 \Rightarrow \varphi_2 \wedge \varphi'_2$.

The final formula we need to test is then of the form

$$\varphi = \varphi_{lab} \wedge \varphi_{x,y} \wedge \psi_1 \wedge \psi_2,$$

where $\varphi_{lab}$ ensures the finite-automata behaviour of labels, and in particular that a final state can be reached, and $\varphi_{x,y}$ asserts the correct behaviour of the variables $x, y$ relative to the labels. It follows that system has a winning strategy for the game with input $\varphi$ if, and only if, there is a positive answer to the reachability problem for the 2-counter machine. Finally, labels can be eliminated by means of special data values encoding *blocks* exactly as done in Section 4.2, and in this way Theorem 3 follows.

# 7 Conclusion

For the decidability result in Section 5, we assumed that in any sub-formula of the form $x \approx \langle\varphi?\rangle^{-1}y$, $\varphi$ is $\top$. We believe that this assumption can possibly be removed if we maintain counters for (variable, formula) pairs instead of variables. We leave the technical details of this extension for future work. An open question

[2] Since in order to make sure that it is the *last* $\perp$ element system has to wait for $\top$ to appear, variable $z$ will change its value at the position immediately after the last $\perp$.

is the decidability status of single-sided games with future obligations restricted to only two data variables; the reduction we have in Section 6 needs three.

Some future directions for research on this topic include finding restrictions other than single-sidedness to get decidability. For the decidable cases, the structure of winning strategies can be studied, e.g., whether memory is needed and if yes, how much.

**Acknowledgements** The authors thank Stéphane Demri and Prakash Saivasan for useful discussions.

# References


[1] P. A. Abdulla, R. Mayr, A. Sangnier, and J. Sproston. Solving parity games on integer vectors. In *CONCUR 2013*, volume 8052 of *LNCS*, pages 106–120. Springer Berlin Heidelberg, 2013.

[2] Parosh Aziz Abdulla, Ahmed Bouajjani, and Julien D'orso. Monotonic and downward closed games. *Journal of Logic and Computation*, 18(1):153–169, 2008.

[3] S. Abriola, D. Figueira, and S. Figueira. Logics of repeating values on data trees and branching counter systems. In *FOSSACS'17*, volume 10203 of *Lecture Notes in Computer Science*, pages 196–212, 2017.

[4] Béatrice Bérard, Serge Haddad, Mathieu Sassolas, and Nathalie Sznajder. Concurrent games on VASS with inhibition. In Maciej Koutny and Irek Ulidowski, editors, *Proceedings of the 23rd International Conference on Concurrency Theory (CONCUR'12)*, volume 7454 of *Lecture Notes in Computer Science*, pages 39–52, Newcastle, UK, September 2012. Springer.

[5] M. Bojańczyk, C. David, A. Muscholl, Th. Schwentick, and L. Segoufin. Two-variable logic on data words. *ACM Transactions on Computational Logic*, 12(4):27, 2011.

[6] Tomáš Brázdil, Petr Jančar, and Antonín Kučera. *Reachability Games on Extended Vector Addition Systems with States*, pages 478–489. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[7] Krishnendu Chatterjee, Mickael Randour, and Jean-François Raskin. Strategy synthesis for multi-dimensional quantitative objectives. *Acta Informatica*, 51(3):129–163, June 2014.

[8] A. Church. Logic, arithmetic and automata. In *Proceedings of the international congress of mathematicians*, volume 1962, pages 23–35, 1962.

[9] T. Colcombet, M. Jurdziński, R. Lazić, and S. Schmitz. Perfect half space games. In *LICS 2017*, pages 1–11, 2017.

[10] S. Demri, D. D'Souza, and R. Gascon. Temporal logics of repeating values. *Journal of Logic and Computation*, 22(5):1059–1096, 2012.

[11] S. Demri, D. Figueira, and M. Praveen. Reasoning about Data Repetitions with Counter Systems. *Logical Methods in Computer Science*, Volume 12, Issue 3:1–55, Aug 2016.

[12] S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. *ACM Transactions on Computational Logic*, 10(3), 2009.

[13] D. Figueira. A decidable two-way logic on data words. In *LICS'11*, pages 365–374. IEEE, 2011.

[14] A. Kara, Th. Schwentick, and Th. Zeume. Temporal logics on words with multiple data values. In *FST&TCS'10*, pages 481–492. LZI, 2010.

[15] M. L. Minsky. Recursive unsolvability of post's problem of 'tag' and other topics in the theory of turing machines. *Annals of Mathematics*, 74:437–455, 1961.

[16] F. Neven, T. Schwentick, and V. Vianu. Finite state machines for strings over infinite alphabets. *ACM Transactions on Computational Logic*, 5(3):403–435, 2004.

[17] A. Pnueli and R. Rosner. On the synthesis of an asynchronous reactive module. *Automata, Languages and Programming*, pages 652–671, 1989.

[18] J.-F. Raskin, M. Samuelides, and L. Van Begin. Games for counting abstractions. *Electronic Notes in Theoretical Computer Science*, 128(6):69 – 85, 2005. Proceedings of the Fouth International Workshop on Automated Verification of Critical Systems (AVoCS 2004).

[19] L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In *CSL'06*, volume 4207 of *Lecture Notes in Computer Science*, pages 41–57. Springer, 2006.

# A Appendix to Section 4

## A.1 Proof details of Lemma 1.1

To simulate counters $c_x$ and $c_y$, we use the environment variable $x$ and system variable $y$. Suppose $\sigma$ is the concrete model built during a game. The value of counter $c_x$ (resp. $c_y$) before the $i^{\text{th}}$ transition is the cardinality of the set $\{d \in \mathbb{D} \mid \exists j \in \{1,\ldots,i\} : \sigma(j)(x) = d, \forall j' \in \{1,\ldots,i\} : \sigma(j')(y) \neq d\}$ (resp. $\{d \in \mathbb{D} \mid \exists j \in \{1,\ldots,i\}, \sigma(j)(y) = d, \forall j' \in \{1,\ldots,i\}, \sigma(j')(x) \neq d\}$). Intuitively, the value of counter $c_x$ is the number of data values that have appeared under variable $x$ but not under $y$. To increment $c_x$, a fresh new data value is assigned to $x$ and the data value assigned to $y$ should be one that has already appeared before under $x$ and $y$. To decrement $c_x$, the same data value should be assigned to $x$ and $y$ and it should have appeared before under $x$ but not under $y$. In order to test that $c_x$ has the value zero, the same data value should be assigned to $x$ and $y$ and it should be one that has already appeared under $x$ and under $y$. In addition, every increment for $c_x$ should have been matched by a subsequent decrement for $c_x$. The operations for $c_y$ should follow similar rules, with $x$ and $y$ interchanged.

For every transition $t$ of the 2-counter machine, there is a Boolean variable $p_t$ owned by $\mathsf{environment}$. The $i^{\text{th}}$ transition chosen by $\mathsf{environment}$ is in the $(i+1)^{\text{st}}$ valuation. The set $\Phi^e$ consists of the following formulas, each of which denotes a mistake made by $\mathsf{environment}$.

- $\mathsf{environment}$ chooses some transition in the first position.

$$\bigvee_{t \text{ is any transition}} p_t$$

- The first transition is not an initial transition.

$$\mathsf{X}(\bigvee_{t \text{ is not an initial transition}} p_t)$$

- $\mathsf{environment}$ chooses more or less than one transition.

$$\mathsf{X}F(\bigvee_{t \neq t'} (p_t \wedge p_{t'}) \vee \bigwedge_{t \text{ is any transition}} (\neg p_t))$$

- Consecutive transitions are not compatible.

$$\mathsf{X}F(\bigvee_{t' \text{ cannot come after } t} (p_t \wedge \mathsf{X}p_{t'}))$$

- A transition increments $x$ but the data value for $x$ is old.

$$\mathsf{X}F(\bigvee_{t \text{ increments } x} p_t \wedge (x \approx \Diamond^{-1}x \vee x \approx \Diamond^{-1}y))$$

- A transition decrements $x$ but the data value for $x$ has not appeared under $x$ or it has appeared under $y$.

$$\mathsf{X}F(\bigvee_{t \text{ decrements } x} p_t \wedge (\neg x \approx \Diamond^{-1}x \vee x \approx \Diamond^{-1}y))$$

- A transition increments $c_y$ but the data value for $x$ is new.

$$\mathsf{X}F(\bigvee_{t \text{ increments } y} p_t \wedge (\neg x \approx \Diamond^{-1}x \vee \neg x \approx \Diamond^{-1}y))$$

- A transition decrements $c_y$ but the data value for $x$ has not appeared before in $y$ or it has appeared before in $x$.

$$\mathsf{X}F(\bigwedge_{t \text{ decrements } y} p_t \wedge (\neg(x \approx \Diamond^{-1}y) \vee x \approx \Diamond^{-1}x))$$

- A transition tests that the value in the counter $c_x$ is zero, but there is a data value that has appeared under $x$ but not under $y$. In such a case, system can map that value to $y$, make the following formula true and win immediately.

$$\mathsf{X}F(\bigvee_{t \text{ tests } c_x \text{ for } 0} p_t \wedge y \approx \Diamond^{-1}x \wedge \neg y \approx \Diamond^{-1}y)$$

- A transition tests that the value in the counter $c_y$ is zero, but there is a data value that has appeared under $y$ but not under $x$. In such a case, system can map that value to $y$, make the following formula true and win immediately.

$$\mathsf{X}F(\bigvee_{t \text{ tests } c_y \text{ for } 0} p_t \wedge y \approx \Diamond^{-1}y \wedge \neg y \approx \Diamond^{-1}x)$$

The set $\Phi^s$ consists of the following formulas, each of which denotes constraints that system has to satisfy after environment makes a move.

- If a transition increments $c_x$, then the data value of $y$ must already have appeared in the past.

$$\mathsf{X}G(\bigwedge_{t \text{ increments } c_x} (p_t \Rightarrow (y \approx \Diamond^{-1}x \wedge y \approx \Diamond^{-1}y)))$$

- If a transition increments $c_y$, then the data value of $y$ must be a fresh one.

$$\mathsf{X}G(\bigwedge_{t \text{ increments } c_y} p_t \Rightarrow \neg(y \approx x) \wedge \neg(y \approx \Diamond^{-1}x) \wedge \neg(y \approx \Diamond^{-1}y))$$

- If a transition decrements $c_x$ or $c_y$ or tests one of them for zero, then the data value of $y$ must be equal to that of $x$.

$$\mathsf{X}G(\bigwedge_{t \text{ decrements/zero tests some counter}} p_t \Rightarrow y \approx x)$$

- The halting state is reached.

$$\mathsf{X}F \bigvee_{t \text{ is a transition whose traget state is halting}} p_t$$

The winning condition of the LRV[$\top$, $\approx$, $\leftarrow$] game is given by the formula $\bigvee \Phi^e \vee \bigwedge \varphi^s$. For system to win, one of the formulas in $\Phi^e$ must be true or all the formulas in $\Phi^s$ must be true. Hence, for system to win, environment should make a mistake during simulation or no one makes any mistake and the halting state is reached. Hence, system has a winning strategy iff the 2-counter machine reaches the halting state.

## A.2 Proof details of Lemma 1.2

Indeed, note that assuming this encoding, we can make sure that we are standing at the left boundary of a block using the LRV formula

$$\varphi_{block(0)} = (x \approx \mathsf{X}^{m+3}x) \wedge x \approx \mathsf{X}x;$$

and we can then test that we are in position $i \in \{1, \ldots, m+2\}$ of a block through the formula

$$\varphi_{block(i)} = \mathsf{X}^{-i}\varphi_{block(0)}.$$

For any fixed linear order on the set of labels $\lambda_1 < \cdots < \lambda_{m-1}$, we will encode that the current block has the $i$-th label $\lambda_i$ as

$$\varphi_{\lambda_i} = \varphi_{block(0)} \wedge \mathsf{X}^2(x \approx \mathsf{X}^i x).$$

Notice that in this coding of labels, a block could have several labels, but of course this is not a problem, if need be one can ensure that exactly one label holds at each block.

$$\varphi_{\textit{1-label}} = \bigvee_{i} \varphi_{\lambda_i} \wedge \neg \bigwedge_{i \neq j} \varphi_{\lambda_i} \wedge \varphi_{\lambda_j}$$

Also, observe that through a binary encoding one could encode the labels in blocks of logarithmic length.

Now the question is: How do we enforce this shape of data words?

Firstly, the structure of a block on variable $x$ can be enforced through the following formula

$$\begin{aligned}\varphi_{\textit{block-str}} = \;& \mathsf{X}^2(\neg(x \approx \Diamond^{-1} x)) \wedge \\ & \bigwedge_{1 < i \leq m+1} \mathsf{X}^i((x \approx \mathsf{X}^{1-i} x) \vee \neg(x \approx \Diamond^{-1} x)) \wedge \\ & \varphi_{\textit{1-label}} \wedge \mathsf{X}^{m+1}(x \approx \mathsf{X} x).\end{aligned}$$

The first two lines ensure that the data values of each $w_i$ are 'fresh' (*i.e.*, they have not appeared before the current block); while the last line ensures that the two last positions repeat the data value and that each blocks encodes exactly one label. Further, a formula can inductively enforce that this structure is repeated on variable $x$:

8) The first position verifies $\varphi_{\textit{block}(0)}$; and for every position we have $\varphi_{\textit{block}(0)} \Rightarrow \varphi_{\textit{block-str}} \wedge \mathsf{X}^{m+2}\varphi_{\textit{block}(0)}$

And secondly, we can make sure that the $y$ variable must have the same data value as the $x$ variable in all positions —except, of course, the $(m+3)$-rd positions of blocks. This can be enforced by making false the formula as soon as the following property holds.

F) There is some $i \in \{0, \ldots, m+1\}$ and some position verifying

$$\varphi_{\textit{block}(i)} \wedge \neg(x \approx y).$$

In each of the formulas $\varphi$ described in the previous section, consider now guarding all positions with $\varphi_{\textit{block}(m+2)}$[3]; replacing each test for a label $\lambda_i$ with $\mathsf{X}^{-(m+2)}\varphi_{\lambda_i}$; and replacing each $\mathsf{X}^i$ with $\mathsf{X}^{(m+3)i}$, obtaining a new formula $\hat{\varphi}$ that works over the block structure encoding we have just described.

Then, for the resulting formula $\bigvee \hat{\Phi}^e \vee \bigwedge \hat{\Phi}^s$ there is a winning strategy for $\mathsf{system}$ if, and only if, there is an accepting run of the 2-counter machine.

## A.3 Proof details of Lemma 1.3

In the reduction from Lemma 1.2, one can use the following encoding to replace testing whether two positions at distance $n$ have the same data value with a chained series of tests. Each block of the form

$$\begin{array}{lcccccccccc} x: & d & d & d_1 & d_2 & d_3 & d_4 & \cdots & d_m & x_i & d & d \\ y: & d & d & d_1 & d_2 & d_3 & d_4 & \cdots & d_m & y_i & d & d \end{array}$$

will now look like:

$$\begin{array}{lcccccccccccccccc} x: & d & d & d_1 & d_2 & d & d_1 & d_3 & d & d_1 & d_4 & \cdots & d & d_1 & d_m & x_i & d & d \\ y: & d & d & d_1 & d_2 & d & d_1 & d_3 & d & d_1 & d_4 & \cdots & d & d_1 & d_m & y_i & d & d \end{array}$$

Note that $d, d_1$ repeats along the whole block in such a way that testing if $d_i = d_1$ and ensuring that the first two data values are equal to the last two data values can be done using data tests of bounded $\mathsf{X}$-distance. Therefore, determining the winner of a LRV game is still undecidable if both the variables and the $\mathsf{X}$-distance is bounded.

[3] That is, where it said "there exists a position where $\psi$ holds", now it should say "there exists a position where $\varphi_{\textit{block}(m+2)} \wedge \psi$ holds", where it said "for every position $\psi$ holds" it should now say "for every position $\varphi_{\textit{block}(m+2)} \Rightarrow \psi$ holds".

# B Appendix to Section 5

## B.1 Proof details for the forward direction of Lemma 1.5

Suppose the system player has a strategy $ts : \Upsilon^* \cdot \Upsilon^e \to \Upsilon^s$ in the single-sided LRV[$\top$, $\approx$, $\leftarrow$] game. We will show that the system player has a strategy $ss : (Q \times \mathbb{N}^C)^* \cdot (Q^s \times \mathbb{N}^C) \to T$ in the single-sided VASS game. It is routine to construct such a strategy from the mapping $\mu : (\mathcal{P}(BVARS^{\varphi}))^* \to \mathsf{FR} \cup \{\perp\}$ that we define now. For every sequence $\chi \in (\mathcal{P}(BVARS^{\varphi}))^*$, we will define $\mu(\chi)$ and a concrete model of length $|\chi|$, by induction on $|\chi|$. For the base case $|\chi| = 0$, the concrete model is the empty sequence and the frame is $\perp$.

For the induction step, suppose $\chi$ is of the form $\chi' \cdot V$ and $\sigma$ is the concrete model defined for $\chi'$ by induction hypothesis. Let $\upsilon^e : BVARS^e \to \{\top, \perp\}$ be the mapping defined as $\upsilon^e(p) = \top$ iff $p \in V$. The system player's strategy $ts$ in the single-sided LRV[$\top$, $\approx$, $\leftarrow$] game will give a valuation $ts(\sigma \cdot \upsilon^e) = \upsilon^s : DVARS^s \to \mathbb{D}$. We define the finite concrete model to be $\sigma \cdot (\upsilon^e \oplus \upsilon^s)$ and $\mu(\chi)$ to be the frame $fr' = \{\varphi' \in \Omega_l^{\varphi} \mid \sigma \cdot (\upsilon^e \oplus \upsilon^s), |\sigma| + 1 - \lceil |\sigma| \rceil l \models \varphi'\}$.

Next we will prove that the strategy $ss$ defined above is winning for the system player. Suppose the system player plays according to $ss$ in the single-sided VASS game, resulting in the sequence of states

$$
\begin{aligned}
&(-1, q^{\varphi}_{init}, \perp)(-1, q^{\varphi}_{init}, \perp, V_1)(0, q^{\varphi}_{init}, fr_1)(0, q^{\varphi}_{init}, fr_1, V_2)\\
&(1, q^{\varphi}_{init}, fr_2) \cdots (l, q, fr_{l+1})(l, q, fr_{l+1}, V_{l+2})(l, q', fr_{l+2}) \cdots
\end{aligned}
$$

The sequence $fr_{l+1} fr_{l+2} \cdots$ is an infinite $(l, \varphi)$-symbolic model; call it $\rho$. It is clear from the construction that $\rho$ is realized by a concrete model $\sigma$, which is the result of the system player playing according to the winning strategy $ts$ in the LRV[$\top$, $\approx$, $\leftarrow$] game. So $\sigma, 1 \models \varphi$ and by Lemma 1.4 (symbolic vs. concrete models), $\rho$ symbolically satisfies $\varphi$. By definition of $A^{\varphi}$, the play satisfies the parity condition in the single-sided VASS game. It remains to prove that if a transition given by $ss$ decrements some counter, that counter will have sufficiently high value. Any play starts with all counters having zero and a counter is decremented by a transition if the frame chosen by that transition has points of decrement for the counter. For $e \in \{1, \ldots, l+1\}$ and $x \in DVARS^{\varphi}$, $[(x, e)]_{fr_e}$ cannot be a point of decrement in $fr_e$ — if it were, the data value $\sigma(e)(x)$ would have appeared in some position in $\{1, \ldots, e-1\}$, creating a backward reference from $(x, e)$ in $fr_e$.

For $i > l+1$, $x \in DVARS^{\varphi}$ and $X \in \mathcal{P}^+(DVARS^{\varphi})$, suppose $[(x, l)]_{fr_i}$ is a point of decrement for $X$ in $fr_i$. Before decrementing the counter $X$, it is incremented for every point of increment for $X$ in every frame $fr_j$ for all $j < i$. Hence, it suffices to associate with this point of decrement a point of increment for $X$ in a frame earlier than $fr_i$ that is not associated to any other point of decrement. Since $[(x, l)]_{fr_i}$ is a point of decrement for $X$ in $fr_i$, the data value $\sigma(i)(x)$ appears in some of the positions $\{1, \ldots, i-l-1\}$. Let $i' = \max\{j \in \{1, \ldots, i-l-1\} \mid \exists y \in DVARS^{\varphi}, \sigma(j)(y) = \sigma(i)(x)\}$. Let $x' \in X$ be such that $\sigma(i')(x') = \sigma(i)(x)$ and associate with $[(x, l)]_{fr_i}$ the class $[(x', 0)]_{fr_{i'+l}}$, which is a point of increment for $X$ in $fr_{i'+l}$. The class $[(x', 0)]_{fr_{i'+l}}$ cannot be associated with any other point of decrement for $X$ — suppose it were associated with $[(y, l)]_{fr_j}$, which is a point of decrement for $X$ in $fr_j$. Then $\sigma(j)(y) = \sigma(i)(x)$. If $j = i$, then $[(x, l)]_{fr_i} = [(y, l)]_{fr_j}$ and the two points of decrement are the same. So $j < i$ or $j > i$. We compute $j'$ for $[(y, l)]_{fr_j}$ with $j' < j$ just like we computed $i'$ for $[(x, l)]_{fr_i}$. If $j < i$, then $j$ would be one of the positions in $\{1, \ldots, i-l-1\}$ where the data value $\sigma(i)(x)$ appears ($j$ cannot be in the interval $[i-l, i-1]$ since those positions do not contain the data value $\sigma(i)(x)$; if they did, there would have been a backward reference from $(x, l)$ in $fr_i$ and $[(x, l)]_{fr_i}$ would not have been a point of decrement), so $j \leq i'$ (and hence $j' < i'$). If $j > i$, then $i$ is one of the positions in $\{1, \ldots, j-l-1\}$ where the data value $\sigma(j)(y)$ appears ($i$ cannot be in the interval $[j-l, j-1]$ since those positions do not contain the data value $\sigma(j)(y)$; if they did, there would have been a backward reference from $(y, l)$ in $fr_j$ and $[(y, l)]_{fr_j}$ would not have been a point of decrement), so $i \leq j'$ (and hence $i' < j'$). In both cases, $j' \neq i'$ and hence, the class $[y', 0]_{fr_{j'+l}}$ we associate with $[(y, l)]_{fr_j}$ would be different from $[(x', 0)]_{fr_{i'+l}}$.

## B.2 Proof details for the reverse direction of Lemma 1.5

Suppose the system player has a strategy $ss : (Q \times \mathbb{N}^C)^* \cdot (Q^s \times \mathbb{N}^C) \to T$ in the single-sided VASS game. We will show that the system player has a strategy $ts : \Upsilon^* \cdot \Upsilon^e \to \Upsilon^s$ in the single-sided LRV[$\top$, $\approx$, $\leftarrow$] game. For every $\sigma \in \Upsilon^*$ and every $\upsilon^e \in \Upsilon^e$, we will define $ts(\sigma \cdot \upsilon^e) : DVARS^{\varphi} \to \mathbb{D}$ and a sequence of configurations $\chi \cdot ((e, q, fr), \vec{n}_{inc} - \vec{n}_{dec})$ in $(Q \times \mathbb{N}^C)^* \cdot (Q^e \times \mathbb{N}^C)$ of length $2|\sigma| + 3$ such that for every counter $X \in \mathcal{P}^+(DVARS^\varphi)$, $\vec{n}_{inc}(X)$ is the sum of the number of points of increment for $X$ in all the frames occurring in $\chi$ and $\vec{n}_{dec}(X)$ is the sum of the number of points of decrement for $X$ in all the frames occurring in $\chi$ and in $fr$. We will do this by induction on $|\sigma|$ and prove that the resulting strategy is winning for the system player. By *frames occurring in* $\chi$, we refer to frames $fr$ such that there are consecutive configurations $((e,q,fr),\vec{n})((e,q,fr,V),\vec{n})$ in $\chi$. By $\Pi_{\mathsf{FR}}(\chi)(i)$, we refer to $i^{\text{th}}$ such occurrence of a frame in $\chi$. Let $\{d_0, d_1, \ldots\} \subseteq \mathbb{D}$ be a countably infinite set of data values.

For the base case $|\sigma| = 0$, let $V \subseteq BVARS^e$ be defined as $p \in V$ iff $v^e(p) = \top$. Let $ss(((-1, q^\varphi_{init}, \perp), \vec{0}) \cdot ((-1, q^\varphi_{init}, \perp, V), \vec{0}))$ be the transition $(-1, q^\varphi_{init}, \perp, V) \xrightarrow{\vec{0}-dec(fr_1)} (0, q, fr_1)$. Since $ss$ is a winning strategy for $\mathsf{system}$ in the single-sided VASS game, $dec(fr_1)$ is necessarily equal to $\vec{0}$. The set of variables $DVARS^\varphi$ is partitioned into equivalence classes by the $(0,\varphi)$-frame $fr_1$. We define $ts(v^e)$ to be the valuation that assigns to each such equivalence class a data value $d_j$, where $j$ is the smallest number such that $d_j$ is not assigned to any variable yet. We let the sequence of configurations be $((-1, q^\varphi_{init}, \perp), \vec{0}) \cdot ((-1, q^\varphi_{init}, \perp, V), \vec{0}) \cdot ((0, q, fr_1), -dec(fr_1))$.

For the induction step, suppose $\sigma \cdot v^e = \sigma' \cdot (v^e_1 \oplus v^s_1) \cdot v^e$ and $\chi' \cdot ((e,q,fr),\vec{n})$ is the sequence of configurations given by the induction hypothesis for $\sigma' \cdot v^e_1$. If $\{\varphi' \in \Omega^\varphi_l \mid \sigma' \cdot (v^e_1 \oplus v^s_1), |\sigma'| + 1 - e \models \varphi'\} \neq fr$, it corresponds to the case where the system player in the LRV$[\top, \approx, \leftarrow]$ game has already deviated from the strategy we have defined so far. So in this case, we define $ts(\sigma \cdot v^e)$ and the sequence of configurations to be arbitrary. Otherwise, we have $\{\varphi' \in \Omega^\varphi_l \mid \sigma' \cdot (v^e_1 \oplus v^s_1), |\sigma'| + 1 - e \models \varphi'\} = fr$. Let $V \subseteq BVARS^e$ be defined as $p \in V$ iff $v^e(p) = \top$ and let $ss(\chi' \cdot ((e,q,fr),\vec{n}) \cdot ((e,q,fr,V),\vec{n}))$ be the transition $(e,q,fr,V) \xrightarrow{inc(fr)-dec(fr')} (\lceil e+1 \rceil l, q', fr')$. We define the sequence of configurations as $\chi' \cdot ((e,q,fr),\vec{n}) \cdot ((e,q,fr,V)\vec{n}) \cdot ((\lceil e+1 \rceil l, q', fr'), \vec{n} + inc(fr) - dec(fr'))$. Since $ss$ is a winning strategy for the system player in the single-sided VASS game, $\vec{n} + inc(fr) - dec(fr') \geq \vec{0}$. The valuation $ts(\sigma \cdot v^e) : DVARS^\varphi \to \mathbb{D}$ is defined as follows. The set $DVARS^\varphi$ is partitioned by the equivalences classes at level $\lceil e+1 \rceil l$ in $fr'$. For every such equivalence class $[(x, \lceil e+1 \rceil l)]_{fr'}$, assign the data value $d'$ as defined below.

1. If there is a backward reference $\mathsf{X}^{\lceil e+1 \rceil l} x \approx \mathsf{X}^{\lceil e+1 \rceil l - j} y$ in $fr'$, let $d' = \sigma' \cdot (v^e_1 \oplus v^s_1)(|\sigma'| + 2 - j)(y)$.

2. If there are no backward references from $(x, \lceil e+1 \rceil l)$ in $fr'$ and the set $\mathsf{PO}_{fr}(x, \lceil e+1 \rceil l)$ of past obligations of $x$ at level $\lceil e+1 \rceil l$ in $fr'$ is empty, let $d'$ be $d_j$, where $j$ is the smallest number such that $d_j$ is not assigned to any variable yet.

3. If there are no backward references from $(x, \lceil e+1 \rceil l)$ in $fr'$ and the set $\mathsf{PO}_{fr}(x, \lceil e+1 \rceil l)$ of past obligations of $x$ at level $\lceil e+1 \rceil l$ in $fr'$ is the non-empty set $X$, then $[(x, \lceil e+1 \rceil l)]_{fr'}$ is a point of decrement for $X$ in $fr'$. Pair off this with a point of increment for $X$ in a frame that occurs in $\chi' \cdot ((e,q,fr),\vec{n}) \cdot ((e,q,fr,V),\vec{n})$ that has not been paired off before. It is possible to do this for every point of decrement for $X$ in $fr'$, since $(\vec{n} + inc(fr))(X)$ is the number of points of increment for $X$ occurring in $\chi' \cdot ((e,q,fr),\vec{n}) \cdot ((e,q,fr,V),\vec{n})$ that have not yet been paired off and $(\vec{n} + inc(fr))(X) \geq dec(fr')(X)$. Suppose we pair off $[(x, \lceil e+1 \rceil l)]_{fr'}$ with a point of increment $[(y,0)]_{fr_i}$ in the frame $fr_i = \Pi_{\mathsf{FR}}(\chi' \cdot ((e,q,fr),\vec{n}) \cdot ((e,q,fr,V),\vec{n}))(i)$, then let $d'$ be $\sigma' \cdot (v^e_1 \oplus v^s_1)(i)(y)$.

Suppose the system player plays according to the strategy $ts$ defined above, resulting in the model $\sigma = (v^e_1 \oplus v^s_1) \cdot (v^e_2 \oplus v^s_2) \cdots$. It is clear from the construction that there is a sequence of configurations

$$\begin{gathered}((-1, q^\varphi_{init}, \perp), \vec{0})((-1, q^\varphi_{init}, \perp, V_1), \vec{0}) \\ ((0, q^\varphi_{init}, fr_1), \vec{n}_1)((0, q^\varphi_{init}, fr_1, V_2), \vec{n}_1) \\ ((1, q^\varphi_{init}, fr_2), \vec{n}_2) \cdots ((l, q, fr_{l+1}), \vec{n}_{l+1}) \\ ((l, q, fr_{l+1}, V_{l+2}), \vec{n}_{l+1})((l, q', fr_{l+2}), \vec{n}_{l+2}) \cdots\end{gathered}$$

that is the result of the system player playing according to the strategy $ss$ in the single-sided VASS game such that the concrete model $\sigma$ realizes the symbolic model $fr_{l+1} fr_{l+2} \cdots$. Since $ss$ is a winning strategy for the system player, the sequence of configurations above satisfy the parity condition of the single-sided VASS game, so $fr_{l+1} fr_{l+2} \cdots$ symbolically satisfies $\varphi$. From Lemma 1.4 (symbolic vs. concrete models), we conclude that $\sigma$ satisfies $\varphi$.

## B.3 Proof details of Theorem 2

Given a single-sided VASS game, we will make the following assumptions about it without loss of generality.

- The initial state belongs to the environment player (if it doesn't, we can add an extra state and a transition to achieve this).
- The environment and system players strictly alternate (if there are transitions between states belonging to the same player, we can add a dummy state belonging to the other player in between).
- The initial counter values are zero (if they aren't, we can add extra transitions before the initial state and force the system player to get the counter values from zero to the required values).

The formula giving the winning condition of the single-sided LRV[$\top$, $\approx$, $\leftarrow$] game is made up of the following variables. Suppose $T^e$ and $T^s$ are the sets of environment and systems transitions repsectively. For every transition $t \in T^e$, there is an environment variable $p_t$. We indicate that the environment player choses a transition $t$ by setting $p_t$ to true. For every transition $t \in T^s$ of the single-sided VASS game, there is a system variable $t$. There is a system variable $ct_s$ to indicate the moves made by the system player. We indicate that the system player choses a transition $t$ by mapping $t$ and $ct_s$ to the same data value. For every counter $x$ of the single-sided VASS game, there are system variables $x$ and $\overline{x}$.

The formula $\varphi_e$ indicates that the environment player makes some wrong move and it is the disjunction of the following formulas.

- The environment does not choose any transition in some round.

$$F( \bigwedge_{t \in T^e} \neg p_t )$$

- The environment choses more than one transition in some round.

$$F( \bigvee_{t \neq t' \in T^e} (p_t \wedge p_{t'}) )$$

- The environment does not start with a transition originating from the designated initial state.

$$\bigvee_{t \in T^e,\ \text{origin of } t \text{ is not the initial state}} p_t$$

- The environment takes some transition that cannot be taken after the previous transition by the system player.

$$\bigvee_{t \in T^s} F( t \approx ct_s \wedge \bigwedge_{t' \in T_s \setminus \{t\}} \neg(t' \approx ct_s) \wedge \bigvee_{t' \in T^e,\ t' \text{ can not come after } t} \mathsf{X}(p_{t'}) )$$

For simulating a counter $x$, we use two variables $x$ and $\overline{x}$. The data values assigned to these variables from positions 1 to $i$ in a concrete model $\sigma$ will represent the counter value that is equal to the cardinality of the set $\{d \in \mathbb{D} \mid \exists j \in \{1, \ldots, i\}, \sigma(j)(x) = d, \forall j' \in \{1, \ldots, i\}, \sigma(j')(\overline{x}) \neq d\}$. We will use special formulas for incrementing, decrementing and retaining previous values of counters.

- To increment a counter represented by $x, \overline{x}$, we force the next data values of $x$ and $\overline{x}$ to be new ones that have never appeared before in $x$ or $\overline{x}$.

$$\varphi_{inc}(x, \overline{x}) = \mathsf{X}\neg( (x \approx \Diamond^{-1} x) \vee (x \approx \Diamond^{-1} \overline{x}) \vee (\overline{x} \approx \Diamond^{-1} x) \vee (\overline{x} \approx \Diamond^{-1} \overline{x}) \vee (x \approx \overline{x}) )$$

- To decrement a counter represented by $x, \overline{x}$, we force the next position to have a data value for $x$ and $\overline{x}$ such that it has appeared in the past for $x$ but not for $\overline{x}$.

$$\varphi_{dec}(x, \overline{x}) = \mathsf{X}(\ x \approx \overline{x} \ \wedge \ x \approx \Diamond^{-1} x \ \wedge \ \neg(x \approx \Diamond^{-1} \overline{x})\ )$$

- To ensure that a counter represented by $x, \overline{x}$ is not changed, we force the next position to have a data value for $x$ that has already appeared in the past for $x$ and we force the next position to have a data value for $\overline{x}$ that has never appeared in the past for $x$ or $\overline{x}$.

$$\varphi_{nc}(x, \overline{x}) = \mathsf{X}(\ x \approx \Diamond^{-1} x \ \wedge \ \neg(\overline{x} \approx \Diamond^{-1} x) \ \wedge \ \neg(\overline{x} \approx \Diamond^{-1} \overline{x})\ )$$

The formula $\varphi_s$ indicates that the system player makes all the right moves and it is the conjunction of the following formulas.

- The system player always choses at least one move.

$$G(\ \bigvee_{t \in T^s} t \approx ct_s\ )$$

- The system player always choses at most one move.

$$G(\ \bigwedge_{t \neq t' \in T^s} \neg(t \approx ct_s \wedge t' \approx ct_s)\ )$$

- The system player always choses a transition that can come after the previous transition chosen by the environment.

$$\bigwedge_{t \in T^e} G(\ p_t \ \Rightarrow \bigvee_{t' \in T_s,\ t' \text{ can come after } t} t' \approx ct_s\ )$$

- The system player sets the initial counter values to zero.

$$\bigwedge_{x \text{ is a counter}} x \approx \overline{x}$$

- The system player updates the counters properly.

$$\begin{aligned} G(\ & \bigwedge_{(q, x++, q') = t \in T^s} (\ t \approx ct_s \ \Rightarrow \ \varphi_{inc}(x, \overline{x}) \ \wedge \bigwedge_{x' \neq x} \varphi_{nc}(x', \overline{x'})) \\ & \bigwedge_{(q, nop, q') = t \in T^s} (\ t \approx ct_s \ \Rightarrow \bigwedge_{x \text{ is a counter}} \varphi_{nc}(x, \overline{x})\ ) \\ & \bigwedge_{(q, x--, q') = t \in T^s} (\ t \approx ct_s \ \Rightarrow \ \varphi_{dec}(x, \overline{x}) \ \wedge \bigwedge_{x' \neq x} \varphi_{nc}(x', \overline{x'}))\ ) \end{aligned}$$

- The maximum colour occuring infinitely often is even.

$$\begin{aligned} \bigvee_{j \text{ is an even colour}} GF( & \bigvee_{t \in T^e, \text{origin of } t \text{ has colour } j} (p_t) \\ \vee & \bigvee_{t \in T^s, \text{origin of } t \text{ has colour } j} (t \approx ct_s)) \ \wedge \\ FG( & \bigwedge_{t \in T,\ \text{origin of } t \text{ has colour greater than } j} \neg(p_t \vee t \approx ct_s)) \end{aligned}$$

The system player wins if the environment player makes any mistake or the system player makes all the moves correctly and satisfies the parity condition. We set the winning condition for the system player in the single-sided LRV[$\top$, $\approx$, $\leftarrow$] game to be $\varphi_e \vee \varphi_s$. If the system player has a winning strategy in the single-sided VASS game, the system player simply makes choices in the single-sided LRV[$\top$, $\approx$, $\leftarrow$] game to imitate the moves in the single-sided VASS game. Since the resulting concrete model satisfies $\varphi_e \vee \varphi_s$, the system player wins. Conversely, suppose the system player has a winning strategy in the single-sided LRV[$\top$, $\approx$, $\leftarrow$] game. In the case where the environment does not make any mistake, the system player has to choose data values such that the simulated sequence of states of the VASS satisfy the parity condition. Hence, the system player in the single-sided VASS game can follow the strategy of the system player in the single-sided LRV[$\top$, $\approx$, $\leftarrow$] game and irrespective of how the environment player plays, the system player wins.

# C Appendix to Section 6

We briefly discuss why the properties $\varphi_{lab}$, $\varphi_{x,y}$, $\psi_1$ and $\psi_2$ can be described in LRV[$\top, \approx, \leftarrow$].

Encoding $\varphi_{lab}$ using some Boolean variables belonging to system is easy since it does not involve the use of data values.

The formula $\varphi_{x,y}$ can be encoded as $\mathsf{G}(\bigwedge_{a \in A} \tau_a \Rightarrow \zeta_a)$ for $A = \{c_x{+}{+}, c_x{-}{-}, c_x = 0?, c_y{+}{+}, c_y{-}{-}, c_y = 0?\}$ and

$$\tau_a = \bigvee_{q,q' \in Q, (q,a,q') \in \delta} \lambda_{(q,a,q')},$$

where $\lambda_{(q,a,q')}$ tests that we are standing on a position labelled with transition $(q, a, q')$ (in particular not a *bis* position). Finally, $\zeta_a$ encodes the rules as already described. That is,

$$\begin{aligned}
\zeta_{c_x++} &= x \approx y \wedge x \approx \mathsf{X}x \wedge y \approx \mathsf{X}y \wedge \neg\mathsf{X}(x \approx \Diamond y),\\
\zeta_{c_x--} &= \neg x \approx y \wedge \neg x \approx \Diamond x \wedge \neg x \approx \Diamond y \wedge\\
&\quad y \approx \mathsf{X}y \wedge y \approx \mathsf{X}x,\\
\zeta_{c_x=0?} &= x \approx y \wedge x \approx \mathsf{X}x \wedge y \approx \mathsf{X}y,
\end{aligned}$$

and similarly for the rules on $c_y$.

The formula $\psi_1 = \mu \wedge ((\pi_1 \wedge \neg\pi'_1) \Rightarrow (\varphi_1 \wedge \neg\varphi'_1))$ is actually composed of two conjunctions $\psi_1 = \psi_1^x \wedge \psi_1^y$, one for $k = x$ and another for $k = y$, let us first suppose that $k = x$. Then,

- $\pi_1$, which checks Property 1, which is simply

$$\pi_1 = b\mathsf{U}(\neg b \wedge \tau_{c_x=0?} \wedge \mathsf{X}(\neg b \mathsf{U} \tau_{c_x--}))$$

- $\pi'_1$, expresses that there exists a $c_x{+}{+}$ with $\neg b$ with value matching that of a future $z$-value:

$$\pi'_1 = \mathsf{F}(\tau_{c_x++} \wedge \neg b \wedge x \approx \Diamond z)$$

- $\varphi_1$, on the other hand, checks that $z$ carries always the same data value, disjoint from the values of all other variables, but as soon as the last $\perp$ value is played the value of $z$ in the next position changes and holds now the $x$ value of that position, and it continues to hold it forever:

$$\begin{aligned}
\varphi_1 = \neg(z \approx \Diamond x \vee z \approx \Diamond y) \wedge&\\
(z \approx \mathsf{X}z)\mathsf{U}\big(\neg(z \approx \mathsf{X}z) \wedge \neg b \wedge \mathsf{X}b \wedge x \approx \mathsf{X}z \wedge \mathsf{XG}(z \approx \mathsf{X}z)\big)&
\end{aligned}$$

- finally, $\varphi'_1$ tests there exists a $c_x{+}{+}$ transition whose principal value corresponds to the $z$-value of some future position:

$$\varphi'_1 = \mathsf{F}(\tau_{c_x++} \wedge x \approx \Diamond z).$$

The formula $\psi_2 = \pi_2 \Rightarrow \varphi_2 \wedge \varphi_2'$ is also composed of two conjuncts, one for $k = x$ and one for $k = y$, let us only show the case $k = x$. Then,

- $\pi_2$ checks Property 2:
$$\pi_2 = b\mathsf{U}(\neg b \wedge \tau_{c_x--} \wedge \mathsf{X}(\mathsf{G}b))$$
- $\varphi_2$ checks chcks that as soon as $b$ turns to $\perp$ hten $z$ at the next position takes the value as current variable $x$, and maintains that value:
$$\begin{aligned}\varphi_2 = \neg(z \approx \Diamond x \vee z \approx \Diamond y) \wedge \\ (z \approx \mathsf{X}z)\mathsf{U}\big(\neg(z \approx \mathsf{X}z) \wedge \neg b \wedge x \approx \mathsf{X}z \wedge \mathsf{XG}(z \approx \mathsf{X}z)\big)\end{aligned}$$
- finally, $\varphi_2'$ tests that in this situation there must be some $c_x++$ position with a data value equal to variable $z$ of a future position:
$$\varphi_2' = \mathsf{F}(\tau_{c_x++} \wedge x \approx \Diamond z).$$

**Correctness.** Suppose first that the 2-counter machine has an accepting run $(q_0, I_1, q_1) \cdots (q_{n-1}, I_n, q_n)$ with $q_n = q_f$. System's strategy is then to play (the encoding of) the labels
$$(q_0, I_1, q_1)\, bis \cdots (q_{n-1}, I_n, q_n)\, bis\, (\hat{t}\, bis)^\omega.$$
In this way, the formula $\varphi_{lab}$ holds.

With respect to the data values on $x, y$, system will respect the rules depicted in Section 6, making $\varphi_{x,y}$ true.

Finally, system will play a data value in $z$ that at the beginning will be some data value which is not used on variables $x$ nor $y$. She will keep this data value all the time, but keeping an eye on the value of $b$ that is being played by environment. If environment plays a first $\perp$ on a $c_k--$ transition, system will then play on $z$, at the next round, the data value of variable $k$ at this round. If environment plays a first $\perp$ at a $c_k = 0?$ transition and a last $\perp$ at a $c_k--$ transition, again system will change the value of $z$ to have the principal value of hte $c_k--$ transition. In this way, system is sure to make true the formula $\varphi_1 \wedge \neg\varphi_1'$ in one case, and formula $\varphi_2 \wedge \varphi_2'$ in the other case. All other cases for $b$ are going to be winning situations for system due to the preconditions $\pi_1 \wedge \neg\pi_1'$ and $\pi_2$ in the formulas $\psi_1$ and $\psi_2$.

On the other hand, if there is no accepting run for the 2-counter machine, then each play of system on variables $x, y$ and the variables verifying both $\varphi_{lab}$ and $\varphi_{x,y}$ must have an illegal transition of type a) or b). At the first illegal transition environment will play $\perp$. If it is a $c_k--$ illegal transition, then environment will continue playing $\top$ in subsequent positons; if it is a $c_k = 0?$ illegal transition, then environment will keep playing $\perp$ until the corresponding $c_k--$ matching to a witnessing $c_k++$ played before the $c_k = 0?$ transition is reached. In either of these situations the antecedent of $\psi_1$ or $\psi_2$ will be true while the consequent will be false; and thus the final formula will not hold, making system uncapable of finding a winning strategy.

Finally, let us explain further how this reduction can be turned into a reduction for the game in which environment plays first and system plays second at each round. For the final formula $\varphi$ of the reduction, let $\varphi'$ be the formula in which environment conditions are shifted one step to the right. This is simply done by replacing every sub-expression of the form $\mathsf{X}^i b$ with $\mathsf{X}^{i+1} b$. It follows that if environment starts playing $\top$ and then continues the play reacting to system strategy in the same way as before, system will have no winning strategy if, and only if, system had no winning strategy in the game where the turns are inverted.