Monadicity of Non-deterministic Logical Matrices is Undecidable

Pedro Filipe Carlos Caleiro Sérgio Marcelino *
SQIG - Instituto de Telecomunicações
Dep. Matemática, Instituto Superior Técnico,
Universidade de Lisboa, Portugal
pedro.g.filipe@tecnico.ulisboa.pt {ccal,smarcel}@math.tecnico.ulisboa.pt

The notion of non-deterministic logical matrix (where connectives are interpreted as multi-functions) preserves many good properties of traditional semantics based on logical matrices (where connectives are interpreted as functions) whilst finitely characterizing a much wider class of logics, and has proven to be decisive in a myriad of recent compositional results in logic. Crucially, when a finite non-deterministic matrix satisfies monadicity (distinct truth-values can be separated by unary formulas) one can automatically produce an axiomatization of the induced logic. Furthermore, the resulting calculi are analytical and enable algorithmic proof-search and symbolic counter-model generation.

For finite (deterministic) matrices it is well known that checking monadicity is decidable. We show that, in the presence of non-determinism, the property becomes undecidable. As a consequence, we conclude that there is no algorithm for computing the set of all multi-functions expressible in a given finite Nmatrix. The undecidability result is obtained by reduction from the halting problem for deterministic counter machines.

1 Introduction

Logical matrices are arguably the most widespread semantic structures for propositional logics [17, 11]. After Łukasiewicz, a logical matrix consists in an underlying algebra, functionally interpreting logical connectives over a set of truth-values, together with a designated set of truth-values. The logical models (valuations) are obtained by considering homomorphisms from the free-algebra in the matrix similarity type into the algebra, and formulas that hold in the model are the ones that take designated values.

However, in recent years, it has become clear that there are advantages in departing from semantics based on logical matrices, by adopting a non-deterministic generalization of the standard notion. Non-deterministic logical matrices (Nmatrices) were introduced in the beginning of this century by Avron and his collaborators [3, 4], and interpret connectives by multi-functions instead of functions. The central idea is that a connective can non-deterministically pick from a set of possible values instead of its value being completely determined by the input values. Logical semantics based on Nmatrices are very malleable, allowing not only for finite characterizations of logics that do not admit finite semantics based on logical matrices, but also for general recipes for various practical problems in logic [13]. Further, Nmatrices still permit, whenever the underlying logical language is sufficiently expressive, to extend from logical matrices general techniques for effectively producing analytic calculi for the induced logics, over which a series of reasoning activities in a purely symbolic fashion can be automated, including proof-search and counter-model generation [16, 3, 4, 2, 9, 14, 7]. In its simplest form, the sufficient expressiveness requirement mentioned above corresponds to *monadicity* [16, 14, 7]. A Nmatrix is *monadic*

*Research funded by FCT/MCTES through national funds and when applicable co-funded by EU under the project UIDB/50008/2020. The first author acknowledges the grant PD/BD/135513/2018 by FCT, under the LisMath PhD programme.

A. Indrzejczak, M. Zawidzki (Eds): 10th International Conference on Non-Classical Logics. Theory and Applications (NCL 2022)
EPTCS 358, 2022, pp. 55–67, doi:10.4204/EPTCS.358.5

when pairs of distinct truth-values can be separated by unary formulas of the logic. This crucial property is decidable, in a straightforward way, for finite logical matrices, as one can simply compute the set of all unary functions expressible in a given finite matrix. However, the computational character of monadicity with respect to Nmatrices has not been studied before.

In this paper we show that, in fact, monadicity is undecidable for Nmatrices. Our proof is obtained by means of a suitable reduction from the halting problem for counter machines, well known to be undecidable [15]. Several details of the construction are inspired by results about the inclusion of infectious values in Nmatrices [8], and also by undecidability results concerning term-dag-automata (a computational model that bears some interesting connections with Nmatrices) [1]. As a consequence, we conclude that the set of all unary multi-functions expressible in a given finite Nmatrix is not computable.

The paper is organized as follows. In Section 2 we introduce and illustrate our objects of study, namely logical matrices and Nmatrices, the logics they induce, and the monadicity property. Section 3 recalls the counter machine model of computation, and shows how its computations can be encoded into suitable Nmatrices. Finally, Section 4 establishes our main results, namely the undecidability of monadicity for Nmatrices, and as a corollary the uncomputability of expressible multi-functions. We conclude, in Section 5, with a discussion of the importance of the results obtained and several topics for further research.

2 Preliminaries

In this section we recall the notion of logical matrix, non-deterministic matrix, and their associated logics. We also introduce, exemplify and discuss the notion of monadicity.

Matrices, Nmatrices and their logics A signature Σ is a family of connectives indexed by their arity, $\Sigma = \{\Sigma^{(k)} : k \in \mathbb{N}\}$. The set of formulas over Σ based on a set of propositional variables P is denoted by $L_\Sigma(P)$. The set of subformulas (resp, variables) of a formula $\varphi \in L_\Sigma(P)$ is denoted by $\mathsf{Sub}(\varphi)$ (resp.,$\mathsf{Var}(\varphi)$). There are two subsets of $L_\Sigma(P)$ that will be of particular interest to us: the set of closed formulas, denoted by $L_\Sigma(\emptyset)$, and the set of unary (or monadic) formulas, denoted by $L_\Sigma(\{p\})$.

A Σ-Nmatrix, is a tuple $\mathbb{M} = \langle A, \cdot_\mathbb{M}, D\rangle$ where A is the set of truth-values, $D \subseteq A$ is the set of designated truth-values, and for each $\copyright \in \Sigma^{(k)}$, the function $\copyright_\mathbb{M} : A^k \to \wp(A) \setminus \{\emptyset\}$ interprets the connective $\copyright$. A Σ-Nmatrix $\mathbb{M}$ is finite if it contains only a finite number of truth-values and Σ is finite. Clearly, this definition generalizes the usual definition of logical matrix, which is recovered when, for every $\copyright \in \Sigma^{(k)}$ and $a_1, \ldots, a_k \in A$, $\copyright_\mathbb{M}(a_1, \ldots, a_k)$ is a singleton. In this case we will sometimes refer to $\copyright_\mathbb{M}$ simply as a function. A valuation over $\mathbb{M}$ is a function $v : L_\Sigma(P) \to A$, such that, $v(\copyright(\varphi_1, \ldots, \varphi_n)) \in \copyright_\mathbb{M}(v(\varphi_1), \ldots, v(\varphi_n))$ for every $\copyright \in \Sigma^{(k)}$. We use $\mathsf{Val}(\mathbb{M})$ to denote the set of all valuations over $\mathbb{M}$. A valuation $v \in \mathsf{Val}(\mathbb{M})$ is said to satisfy a formula φ if $v(\varphi) \in D$, and is said to falsify φ, otherwise. Note that every formula $\varphi \in L_\Sigma(P)$, with $\mathsf{Var}(\varphi) = \{p_1, \ldots, p_k\}$, defines a multi-function $\varphi_\mathbb{M} : A^k \to \wp(A) \setminus \{\emptyset\}$ as $\varphi_\mathbb{M}(x_1, \ldots, x_k) = \{v(\varphi) : v \in \mathsf{Val}(\mathbb{M}), v(p_i) = x_i, 1 \leq i \leq k\}$. The multi-function $\varphi_\mathbb{M}$ is said to be *represented*, or *expressed*, by the formula φ in $\mathbb{M}$. Furthermore, we say that a multi-function f is *expressible* in an Nmatrix $\mathbb{M}$ if there is a formula φ such that $\varphi_\mathbb{M} = f$.

The logic induced by an Nmatrix $\mathbb{M}$ is the Tarskian consequence relation $\vdash_\mathbb{M} \subseteq \wp(L_\Sigma(P)) \times L_\Sigma(P)$ defined as $\Gamma \vdash_\mathbb{M} \varphi$ whenever, for every $v \in \mathsf{Val}(\mathbb{M})$, if $v(\Gamma) \subseteq D$ then $v(\varphi) \in D$. This definition generalizes the usual logical matrix semantics [17, 11]. As usual, a formula φ is said to be a theorem of $\mathbb{M}$ if $\emptyset \vdash_\mathbb{M} \varphi$.

Monadicity Given a Σ-Nmatrix $\mathbb{M} = \langle A, \cdot_{\mathbb{M}}, D\rangle$, we say that $a, b \in A$ are separated, written $a\#b$ if $a \in D$ and $b \notin D$, or vice-versa. A pair of sets of elements $X, Y \subseteq A$ are separated, written $X\#Y$, if for every $a \in X$ and $b \in Y$ we have that $a\#b$. Note that $X\#Y$ precisely if $X \subseteq D$ and $Y \subseteq A \setminus D$, or vice versa. A monadic formula $\varphi \in L_\Sigma(\{p\})$ such that $\varphi_{\mathbb{M}}(a)\#\varphi_{\mathbb{M}}(b)$ is said to separate a and b. We say that a set of monadic formulas S is a set of monadic separators for $\mathbb{M}$ when, for every pair of distinct truth-values of $\mathbb{M}$, there is a formula of S separating them. An Nmatrix $\mathbb{M}$ satisfies monadicity (or simply, is monadic) if there is a set of monadic separators for $\mathbb{M}$.

Example 2.1. Consider the signature $\Sigma = \{\neg, \vee, \wedge, \rightarrow\}$ and the Σ-matrix $\mathbb{M}_{Ł} = \langle A, \cdot_{Ł}, D\rangle$, with $A = \{0, \frac{1}{2}, 1\}$ and $D = \{1\}$, corresponding to Łukasiewicz 3-valued logic, with interpretations as described in the following tables.

x	$\neg_{Ł}(x)$
0	1
$\frac{1}{2}$	$\frac{1}{2}$
1	0

$\vee_{Ł}$	0	$\frac{1}{2}$	1
0	0	$\frac{1}{2}$	1
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1
1	1	1	1

$\wedge_{Ł}$	0	$\frac{1}{2}$	1
0	0	0	0
$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$
1	0	$\frac{1}{2}$	1

$\rightarrow_{Ł}$	0	$\frac{1}{2}$	1
0	1	1	1
$\frac{1}{2}$	$\frac{1}{2}$	1	1
1	0	$\frac{1}{2}$	1

$\mathbb{M}_{Ł}$ is monadic with $\{p, \neg p\}$ as a set of separators. Indeed, p separates 1 from 0, and also 1 from $\frac{1}{2}$, whereas $\neg p$ separates 0 and $\frac{1}{2}$. One may wonder, though, whether one could separate 0 and $\frac{1}{2}$ without using negation. △

Remark 2.2. Notice that we can decide if a given matrix $\mathbb{M}$ is monadic by algorithmically generating every unary function expressible in $\mathbb{M}$, as it is usually done when calculating clones over finite algebras [12]. This procedure is, however, quite expensive, since there are, at most, n^n unary formulas from a set with n values to itself. The next example illustrates this procedure.

Example 2.3. Let $\mathbb{M}'_{Ł}$ be the $\{\vee, \wedge, \rightarrow\}$-reduct of $\mathbb{M}_{Ł}$ introduced in Example 2.1, corresponding to the negationless fragment of Łukasiewicz 3-valued logic. Let us show that $\mathbb{M}'_{Ł}$ is not monadic, by generating every unary expressible function. For simplicity, we represent a unary function $h : A \rightarrow A$ as a tuple $(h(0), h(\frac{1}{2}), h(1))$.

The formulas p, $p \vee p$ and $p \wedge p$ define the same function $(0, \frac{1}{2}, 1)$ and the formula $p \rightarrow p$ defines the constant function $(1, 1, 1)$. It is easy to see that we cannot obtain new functions by further applying connectives, so $(0, \frac{1}{2}, 1)$ and $(1, 1, 1)$ are the only expressible unary functions. For example, $((p \rightarrow p) \rightarrow (p \rightarrow p))_{\mathbb{M}'_{Ł}} = (1, 1, 1)$. We conclude that 0 cannot be separated from $\frac{1}{2}$, and so $\mathbb{M}'_{Ł}$ is not monadic. △

Monadicity in Nmatrices In the presence of non-determinism, we cannot follow the strategy described in Remark 2.2 and Example 2.3. A fundamental difference from the deterministic case is that the multi-functions represented by formulas in a Nmatrix are sensitive to the syntax since, even if there are multiple choices for the value of a subformula, all its occurrences need to have the same value. Crucially, on an Nmatrix $\mathbb{M}$, the multi-function $\mathbb{C}(\varphi_1, \ldots, \varphi_k)_{\mathbb{M}}$ does not depend only on the multi-functions $\mathbb{C}_{\mathbb{M}}$ and $(\varphi_1)_{\mathbb{M}}, \ldots, (\varphi_k)_{\mathbb{M}}$, as we shall see in the next example.

Hence, contrarily to what happens in the deterministic case, when generating the expressible multi-functions in an Nmatrix $\mathbb{M}$ (to find if $\mathbb{M}$ is monadic, or for any other purpose), we cannot just keep the information about the multi-functions themselves but also about the formulas that produce them. Otherwise, we might generate a non-expressible function (as every occurrence of a subformula must have the same value) or miss some multi-functions that are still expressible.

With the intent of making the notation lighter, when representing the interpretation of the connectives using tables, we drop the curly brackets around the elements of an output set. For example, in the table of Example 2.4 we simply write a, c instead of $\{a, c\}$.

Example 2.4. Consider the signature Σ with only one binary connective g, and two Nmatrices $\mathbb{M} = \langle\{a,b,c\},\cdot_{\mathbb{M}},D\rangle$ and $\mathbb{M}' = \langle\{a,b,c\},\cdot_{\mathbb{M}'},\{c\}\rangle$, with interpretation described in the following table.

$g_{\mathbb{M}}$	a	b	c
a	c	a	b,c
b	b	c	a,c
c	b,c	a,c	c

$g_{\mathbb{M}'}$	a	b	c
a	c	a	c
b	b	c	a
c	b,c	a,c	c

Let $\varphi = g(g(p,p),p)$ and $\psi = g(p,g(p,p))$. In $\mathbb{M}$, $\varphi_{\mathbb{M}} = \psi_{\mathbb{M}} = (\{b,c\},\{a,c\},\{c\})$. Although these formulas define the same multi-function, they are not interchangeable, as $g(\varphi,\varphi)_{\mathbb{M}} = g(\psi,\psi)_{\mathbb{M}} = g(p,p)_{\mathbb{M}} = (\{c\},\{c\},\{c\})$ but $g(\varphi,\psi)_{\mathbb{M}} = g(\psi,\varphi)_{\mathbb{M}} = (\{a,c\},\{b,c\},\{c\})$, thus illustrating the already mentioned sensitivity to the syntax. Still, consider $v_x : L_\Sigma(P) \to \{a,b,c\}$, with $x \in \{a,b,c\}$, defined as

$$v_x(\gamma) = \begin{cases} x & \text{if } \gamma \in P \\ c & \text{otherwise.} \end{cases}$$

These functions can easily be shown to be valuations over $\mathbb{M}$ for every choice of x and so, for every unary formula $\varphi \neq p$, we have that $c \in \varphi_{\mathbb{M}}(a) \cap \varphi_{\mathbb{M}}(b) \cap \varphi_{\mathbb{M}}(c)$. We conclude that, apart from p, no unary formula can separate elements of $\{a,b,c\}$, and so $\mathbb{M}$ is not monadic, for any choice of D.

In $\mathbb{M}'$ the outcome is radically different. As $g(p,p)_{\mathbb{M}'}(x) = \{c\}$ for every $x \in \{a,b,c\}$, we have $g(p,g(p,p))_{\mathbb{M}'}(a) = \{c\}$ and $g(p,g(p,p))_{\mathbb{M}'}(b) = \{a\}$ and so, in this case, $\mathbb{M}'$ is monadic with set of separators $\{p, g(p,g(p,p))\}$. △

3 Counter machines and Nmatrices

In this section we recall the essentials of counter machines, and define a suitable finite Nmatrix representing the computations of any given counter machine.

Counter machines

A *(deterministic) counter machine* is a tuple $\mathscr{C} = \langle n, Q, q_{\mathsf{init}}, \delta\rangle$, where $n \in \mathbb{N}$ is the number of counters, Q is a finite set of states, $q_{\mathsf{init}} \in Q$ is the initial state, and δ is a partial transition function $\delta : Q \nrightarrow (\{i^{++} : 1 \leq i \leq n\} \times Q) \cup (\{i^{\mathsf{test}} : 1 \leq i \leq n\} \times Q^2)$.

The set of halting states of $\mathscr{C}$ is denoted by $H = \{q \in Q : \delta(q) \text{ is undefined}\}$.

A configuration of $\mathscr{C}$ is a tuple $C = (q,\vec{x}) \in Q \times \mathbb{N}_0^n$, where q is a state, and $\vec{x} = x_1,\ldots,x_n$ are the values of the counters. Let $\mathsf{Conf}(\mathscr{C})$ be the set of all configurations. $C \in \mathsf{Conf}(\mathscr{C})$ is said to be the initial configuration if $q = q_{\mathsf{init}}$ and $\vec{x} = \vec{0}$. C is said to be a halting configuration if $q \in H$.

When $(q,\vec{y})$ is not a halting configuration, the transition function δ completely determines the next configuration $\mathsf{nxt}(q,\vec{y})$ as follows:

$$\mathsf{nxt}(q,\vec{y}) = \begin{cases} (q',\vec{y}+\vec{\mathsf{e}}_i) & \text{if } \delta(q) = \langle i^{++},q'\rangle, \\ (q'',\vec{y}) & \text{if } \delta(q) = \langle i^{\mathsf{test}},q'',q'''\rangle \text{ and } x_i = 0, \\ (q''',\vec{y}-\vec{\mathsf{e}}_i) & \text{if } \delta(q) = \langle i^{\mathsf{test}},q'',q'''\rangle \text{ and } x_i \neq 0, \end{cases}$$

where $\vec{\mathsf{e}}_i$ is such that $(\mathsf{e}_i)_i = 1$ and $(\mathsf{e}_i)_j = 0$, for all $j \neq i$.

The computation of $\mathscr{C}$ is a finite or infinite sequence of configurations $\langle C_i\rangle_{i<\eta}$, where $\eta \in \mathbb{N}_0 \cup \{\omega\}$ such that C_0 is the initial configuration, and for each $i < \eta$, either C_i is a halting configuration and $i+1 = \eta$ is the length of the computation, or else $C_{i+1} = \mathsf{nxt}(C_i)$.

The intuition behind the transitions of a counter machine is clear from the underlying notion of computation, and in particular the definition of the next configuration. Clearly, $\delta(q) = \langle i^{++}, r\rangle$ results in incrementing the i-th counter and moving to state r, whereas $\delta(q) = \langle i^{\mathsf{test}}, r, s\rangle$ either moves to state r, leaving the counters unchanged, when the value of the i-th counter is zero, or moves to state s, and decrements the i-th counter, when its value is not zero. As usual in counter machine models (see [15]), and also for the sake of simplicity, we are assuming that in the initial configuration all counters have value 0. This is well known not to hinder the computational power of the model, as a machine can always start by setting the counters to other desired input values. We will base our undecidability result on the following well known result about counter machines.

Theorem 3.1 ([15])**.** It is undecidable if a given counter machine halts when starting with all counters set to zero.

In what follows, given $\vec{x} \in A^n$ and $f : A \to B$, we define $f(\vec{x}) \in B^n$ as $f(\vec{x}) = \langle f(x_i) : 1 \leq i \leq n\rangle$. For a given counter machine $\mathscr{C} = \langle n, Q, q_{\mathsf{init}}, \delta\rangle$, we define the signature $\Sigma_{\mathscr{C}}$ such that $\Sigma^{(0)}_{\mathscr{C}} = \{\mathsf{zero}, \varepsilon\}$, $\Sigma^{(1)}_{\mathscr{C}} = \{\mathsf{suc}\}$, $\Sigma^{(n+1)}_{\mathscr{C}} = \{\mathsf{step}_q : q \in Q\}$ and $\Sigma^{(j)}_{\mathscr{C}} = \emptyset$, for all $j \notin \{0, 1, n+1\}$. As usual, zero and suc allow us to encode every $k \in \mathbb{N}_0$ as the closed formula $\mathsf{enc}(k) = \mathsf{suc}^k(\mathsf{zero})$. Moreover, we can encode every finite sequence of configurations $\langle C_0, \ldots, C_k\rangle$ as a sequence formula in the following way:

- $\mathsf{seq}(\langle\rangle) = \varepsilon$, and $\mathsf{seq}(\langle C_0, \ldots, C_{k-1}, (q, \vec{y})\rangle) = \mathsf{step}_q(\mathsf{seq}(\langle C_0, \ldots, C_{k-1}\rangle), \mathsf{enc}(\vec{y}))$.

We will construct a finite Nmatrix $\mathbb{M}_{\mathscr{C}}$ over $\Sigma_{\mathscr{C}}$ that recognizes as a theorem precisely the finite computation of $\mathscr{C}$, if it exists. This means that $\mathbb{M}_{\mathscr{C}}$ can only falsify a formula φ if it is not a sequence formula, or if $\varphi = \mathsf{seq}(\langle C_0, \ldots, C_k\rangle)$ but C_0 is not the initial configuration of $\mathscr{C}$, or C_k is not a halting configuration of $\mathscr{C}$, or $\mathsf{nxt}(C_i) \neq C_{i+1}$ for some $0 \leq i < k$.

From counter machines to Nmatrices

For a given counter machine $\mathscr{C} = \langle n, Q, q_{\mathsf{init}}, \delta\rangle$ let

$$\mathbf{Rm} = \{\mathtt{r}_{=0}, \mathtt{r}_{\geq 0}, \mathtt{r}_{\geq 1}, \mathtt{r}_{\geq 2}\} \qquad \text{and} \qquad \mathbf{Conf} = \{\mathtt{conf}_{q,\overrightarrow{\mathtt{r}}} : q \in Q, \overrightarrow{\mathtt{r}} \in \mathbf{Rm}^n\}$$

and consider the Nmatrix $\mathbb{M}_{\mathscr{C}} = \langle A, \cdot_{\mathbb{M}_{\mathscr{C}}}, D\rangle$ over $\Sigma_{\mathscr{C}}$, where

$$A = \mathbf{Rm} \cup \mathbf{Conf} \cup \{\mathtt{init}, \mathtt{error}\} \qquad \text{and} \qquad D = \{\mathtt{conf}_{q,\overrightarrow{\mathtt{r}}} : q \in H, \overrightarrow{\mathtt{r}} \in \mathbf{Rm}^n\} \qquad \text{and}$$

$$\mathsf{zero}_{\mathbb{M}_{\mathscr{C}}} = \{\mathtt{r}_{=0}, \mathtt{r}_{\geq 0}\} \qquad \varepsilon_{\mathbb{M}_{\mathscr{C}}} = \{\mathtt{init}\} \qquad \mathsf{suc}_{\mathbb{M}_{\mathscr{C}}}(x) = \begin{cases} \{\mathtt{r}_{\geq 1}\} & \text{if } x = \mathtt{r}_{=0} \\ \{\mathtt{r}_{\geq 0}, \mathtt{r}_{\geq 1}\} & \text{if } x = \mathtt{r}_{\geq 0} \\ \{\mathtt{r}_{\geq 2}\} & \text{if } x \in \{\mathtt{r}_{\geq 1}, \mathtt{r}_{\geq 2}\} \\ \{\mathtt{error}\} & \text{otherwise} \end{cases}$$

$$(\mathsf{step}_q)_{\mathbb{M}_{\mathscr{C}}}(x, \vec{z}) = \begin{cases} \{\mathtt{conf}_{q,\vec{z}}\} & \text{if } x = \mathtt{init}, q = q_{\mathsf{init}} \text{ and } \vec{z} \in \{\mathtt{r}_{=0}\}^n \cup \{\mathtt{r}_{\geq 0}\}^n, \text{ or} \\ & \text{if } x = \mathtt{conf}_{q',\overrightarrow{y}}, \overrightarrow{z} \in \mathbf{Rm}^n, \text{ and} \\ & \quad \delta(q') = \langle i^{\mathsf{test}}, q, s\rangle, y_i \in \{\mathtt{r}_{=0}, \mathtt{r}_{\geq 0}\} \text{ and } \overrightarrow{y} = \overrightarrow{z}, \text{ or} \\ & \quad \delta(q') = \langle i^{\mathsf{test}}, s, q\rangle, y_i \in \mathsf{suc}_{\mathbb{M}_{\mathscr{C}}}(z_i) \text{ and } z_j = y_j \text{ for } j \neq i, \text{ or} \\ & \quad \delta(q') = \langle i^{++}, q\rangle, z_i \in \mathsf{suc}_{\mathbb{M}_{\mathscr{C}}}(y_i) \text{ and } z_j = y_j \text{ for } j \neq i \\ \{\mathtt{error}\} & \text{otherwise} \end{cases}$$

where, $s \in Q$ represents an arbitrary state.

$\mathbb{M}_{\mathscr{C}}$ is conceived as a finite way of representing the behavior of $\mathscr{C}$. For that purpose, it is useful to understand the operations zero and suc as means of representing the natural number values of the counters. Their interpretation, however, is finitely defined over the abstract values **Rm**. In fact, in order to check if some formula φ encodes a sequence of computations respecting nxt, it is not essential to distinguish all natural values. Indeed, it is easy to conclude from the definition of counter machine that in each computation step its counters either retain their previous values, or else they are incremented or decremented. As we set the initial configuration with all counters set to zero and the effect of test transitions also depends on detecting zero values, it is sufficient to being able to characterize unambiguously the value 0 and, additionally, being able to recognize pairs of values whose difference is larger than one. This is successfully accomplished with the proposed non-deterministic interpretation of suc, as shall be made clear below. The ε and step operations are then meant to represent sequences of configurations, whereas their interpretation over the abstract values $\mathbf{Conf} \cup \{\mathtt{init}\}$ guarantees that consecutive configurations respect nxt. Of course, the designated values of $\mathbb{M}_{\mathscr{C}}$ are those corresponding to halting configurations. The $\mathtt{error}$ value is absorbing with respect to the interpretation of all operations, and gathers all meaningless situations. Overall, as we will show, $\mathbb{M}_{\mathscr{C}}$ induces a logic that has at most one theorem, corresponding to the computation of $\mathscr{C}$, if it is halting.

The inner workings of the construction

In the next examples we will illustrate the way the Nmatrix $\mathbb{M}_{\mathscr{C}}$ encodes the computations of $\mathscr{C}$. Proofs of the general statements are postponed to the next section.

In order for $\vdash_{\mathbb{M}_{\mathscr{C}}}$ to have, at most, the formula representing the computation of $\mathscr{C}$ as theorem, $\mathsf{Val}(\mathbb{M}_{\mathscr{C}})$ must contain enough valuations to refute every formula not representing the computation of $\mathscr{C}$. These valuations are presented in the following example

Example 3.2. By definition of seq, it is clear that no formula containing variables corresponds to a sequence of configurations. Furthermore, no formula containing variables can be a theorem of $\mathbb{M}_{\mathscr{C}}$ since these formulas are easily refuted by any valuation sending the variables to the truth value $\mathtt{error}$, as this value is absorbing (aka infectious), that is, $\mathsf{suc}_{\mathbb{M}_{\mathscr{C}}}(x) = \mathtt{error}$ whenever $x = \mathtt{error}$ and $(\mathsf{step}_q)_{\mathbb{M}_{\mathscr{C}}}(x, \vec{z}) = \mathtt{error}$ whenever $x = \mathtt{error}$ or $z_i = \mathtt{error}$. Because of this, from here onwards, we concern ourselves only with the truth-values assigned to closed formulas.

We do not have much freedom left, but it will be enough. The interpretations of the connectives are all deterministic, except in the case of $\mathsf{zero}_{\mathbb{M}_{\mathscr{C}}}$ and $\mathsf{suc}_{\mathbb{M}_{\mathscr{C}}}(\mathtt{r}_{\geq 0})$. This means that, if $v \in \mathsf{Val}(\mathbb{M}_{\mathscr{C}})$ and $v(\mathsf{zero}) = \mathtt{r}_{=0}$, then there is no choice left for the values assigned by v to the remaining closed formulas. Consider, therefore, the following valuation

$$v_0^{=}(\psi) = \begin{cases} \mathtt{r}_{=0} & \text{if } \psi = \mathsf{zero}, \\ \mathtt{r}_{\geq 1} & \text{if } \psi = \mathsf{enc}(1), \\ \mathtt{r}_{\geq 2} & \text{if } \psi = \mathsf{enc}(j) \text{ with } j \geq 2, \\ \mathtt{init} & \text{if } \psi = \varepsilon, \\ (\mathsf{step}_q)_{\mathbb{M}_{\mathscr{C}}}(v_0^{=}(\varphi), \vec{z}) & \text{if } \psi = \mathsf{step}_q(\varphi, \psi_1, \ldots, \psi_n) \text{ and } z_i = v_0^{=}(\psi_i), \\ \mathtt{error}, & \text{otherwise.} \end{cases}$$

If $v(\mathsf{zero}) = \mathtt{r}_{\geq 0}$, however, we can still loop the truth values assigned to formulas of the form $\mathsf{enc}(j)$. The amount of loops could be infinite or finite, though the infinite case is of no interest to us, since it does not allow us to falsify any of the formulas that we want to falsify.

Let $k \in \mathbb{N}_0$, consider the valuations

$$v_k(\psi) = \begin{cases} \mathtt{r}_{\geq 0} & \text{if } \psi = \mathsf{enc}(j) \text{ with } j \leq k, \\ \mathtt{r}_{\geq 1} & \text{if } \psi = \mathsf{enc}(j) \text{ with } j = k+1, \\ \mathtt{r}_{\geq 2} & \text{if } \psi = \mathsf{enc}(j) \text{ with } j \geq k+2, \\ \mathtt{init} & \text{if } \psi = \varepsilon, \\ (\mathsf{step}_q)_{\mathbb{M}_{\mathscr{C}}}(v_k(\varphi), \vec{z}) & \text{if } \psi = \mathsf{step}_q(\varphi, \psi_1, \ldots, \psi_n) \text{ and } z_i = v_k(\psi_i), \\ \mathtt{error}, & \text{otherwise.} \end{cases}$$ △

As previously discussed, it is crucial for the valuations in $\mathsf{Val}(\mathbb{M}_{\mathscr{C}})$ to be able to identify whenever two given numbers $a, b \in \mathbb{N}_0$ are not consecutive, or different. To this aim, for every pair $a, b \in (\mathbb{N}_0)^2$, we denote by $\mu^{+}_{a,b}, \mu^{-}_{a,b}, \mu^{\neq}_{a,b}$ the valuations determined by the following conditions

$$\mu^{+}_{a,b} = \begin{cases} v_a & \text{if } b \geq a+1, \\ v_{a-1} & \text{if } b \leq a \text{ and } a \neq 0, \\ v^{=}_0 & \text{if } b \leq a \text{ and } a = 0. \end{cases} \quad \mu^{-}_{a,b} = \begin{cases} v_{a-2} & \text{if } b \leq a-2, \\ v_{a-1} & \text{if } b \geq a-1 \text{ and } a \neq 0, \\ v^{=}_0 & \text{if } b \geq a-1 \text{ and } a = 0. \end{cases} \quad \mu^{\neq}_{a,b} = \begin{cases} v^{=}_0 & \text{if } a = 0, \\ v_{a-1} & \text{if } a \neq 0. \end{cases}$$

Remark 3.3. The following properties can be easily checked by inspecting the corresponding definition:

- if $b \neq a+1$ then $\mu^{+}_{a,b}(\mathsf{enc}(b)) \notin \mathsf{suc}_{\mathbb{M}_{\mathscr{C}}}(\mu^{+}_{a,b}(\mathsf{enc}(a)))$,
- if $b \neq a-1$ then $\mu^{-}_{a,b}(\mathsf{enc}(a)) \notin \mathsf{suc}_{\mathbb{M}_{\mathscr{C}}}(\mu^{-}_{a,b}(\mathsf{enc}(b)))$, and
- if $b \neq a$ then $\mu^{\neq}_{a,b}(\mathsf{enc}(b)) \neq \mu^{\neq}_{a,b}(\mathsf{enc}(a))$.

In the following two examples we consider two different machines that should make clear the soundness of our construction. In the first one, we show how every valuation validates the formula encoding a finite computation. We also see how sequences of configurations can fail to respect nxt in different ways and how we can use the valuations presented in Example 3.2 to falsify formulas encoding them. In the second example, we show a counter machine that never halts.

Example 3.4. Consider the counter machine $\mathscr{C} = \langle 1, Q, q_{\mathsf{init}}, \delta \rangle$ with $Q = \{q_{\mathsf{init}}, q_1, q_2, q_3\}$ and δ as defined in the following table

q	q_{init}	q_1	q_2	q_3
$\delta(q)$	$\langle 1^{++}, q_1 \rangle$	$\langle 1^{\mathsf{test}}, q_3, q_2 \rangle$	$\langle 1^{\mathsf{test}}, q_3, q_3 \rangle$	undefined

The only halting state of $\mathscr{C}$ is q_3 and the machine $\mathscr{C}$ has the following finite computation

$$\langle (q_{\mathsf{init}}, 0), (q_1, 1), (q_2, 0), (q_3, 0) \rangle.$$

For every $v \in \mathsf{Val}(\mathbb{M}_{\mathscr{C}})$, we have $v(\varepsilon) = \mathtt{init}$. The values of $v(\mathsf{enc}(k))$ are dependent on v: if $v = v^{=}_0$ then $v(\mathsf{enc}(0)) = \mathtt{r}_{=0}$ and $v(\mathsf{enc}(1)) = \mathtt{r}_{\geq 1}$. If $v \neq v^{=}_0$ then $v(\mathsf{enc}(0)) = \mathtt{r}_{\geq 0}$ and $v(\mathsf{enc}(1)) \in (\mathsf{suc})_{\mathbb{M}_{\mathscr{C}}}(\mathtt{r}_{\geq 0}) = \{\mathtt{r}_{\geq 0}, \mathtt{r}_{\geq 1}\}$.

Let φ_j be the formula representing the prefix with only the first $j+1$ configurations, we obtain, from the above equalities and the definition of $\mathsf{step}_{\mathbb{M}}$, that

$$\begin{aligned} v(\varphi_0) &= v(\mathsf{step}_{q_{\mathsf{init}}}(\varepsilon, \mathsf{enc}(0))) = (\mathsf{step}_{q_{\mathsf{init}}})_{\mathbb{M}}(\mathtt{init}, v(\mathsf{enc}(0))) = \mathtt{conf}_{q_{\mathsf{init}}, v(\mathsf{enc}(0))} \\ v(\varphi_1) &= v(\mathsf{step}_{q_1}(\varphi_0, \mathsf{enc}(1))) = (\mathsf{step}_{q_1})_{\mathbb{M}}(\mathtt{conf}_{q_{\mathsf{init}}, v(\mathsf{enc}(0))}, v(\mathsf{enc}(1))) = \mathtt{conf}_{q_1, v(\mathsf{enc}(1))} \\ v(\varphi_2) &= v(\mathsf{step}_{q_2}(\varphi_1, \mathsf{enc}(0))) = (\mathsf{step}_{q_2})_{\mathbb{M}}(\mathtt{conf}_{q_1, v(\mathsf{enc}(1))}, v(\mathsf{enc}(0))) = \mathtt{conf}_{q_2, v(\mathsf{enc}(0))} \\ v(\varphi_3) &= v(\mathsf{step}_{q_3}(\varphi_2, \mathsf{enc}(0))) = (\mathsf{step}_{q_3})_{\mathbb{M}}(\mathtt{conf}_{q_2, v(\mathsf{enc}(0))}, v(\mathsf{enc}(0))) = \mathtt{conf}_{q_3, v(\mathsf{enc}(0))} \end{aligned}$$

The formula φ_3 encodes the finite computation of $\mathscr{C}$ and, since $\mathtt{conf}_{q_3,v(\mathsf{enc}(0))} \in D$, $\emptyset \vdash_{\mathbb{M}_\mathscr{C}} \varphi_3$. Furthermore, the formulas φ_i with $0 \leq i \leq 3$, that encode its strict prefixes, are falsified by all valuations, since q_3 is the only halting state.

Formulas not representing sequences of configurations, like $\mathsf{suc}(\psi)$ with $\psi \neq \mathsf{enc}(j)$, are falsified by every $v \in \mathsf{Val}(\mathbb{M}_\mathscr{C})$ since $v(\mathsf{suc}(\psi)) = \mathsf{suc}_\mathbb{M} = \mathtt{error}$. Formulas encoding sequences of configurations not starting in the initial configuration of $\mathbb{M}$ are also falsifiable: $v_0(\mathsf{step}_q(\psi, \mathsf{enc}(j))) = \mathtt{error}$ whenever, either $q \neq q_\mathsf{init}$ and $\psi = \varepsilon$, or, $q = q_\mathsf{init}$, $\psi = \varepsilon$ and $j \neq 0$. For example, $v_0(\mathsf{step}_{q_\mathsf{init}}(\varepsilon, \mathsf{enc}(1))) = (\mathsf{step}_{q_\mathsf{init}})_{\mathbb{M}_\mathscr{C}}(\mathtt{init}, \mathtt{r}_{\geq 1}) = \mathtt{error}$.

The sequence $\langle (q_\mathsf{init}, 0), (q_1, 2) \rangle$, encoded by $\psi = \mathsf{step}_{q_1}(\varphi_0, \mathsf{enc}(2))$, illustrates a situation where the value in the counter was incremented by two while the transition $\delta(q_1) = \langle 1^{++}, q_2 \rangle$ required it to increase by only one. In this case, we have $\mu^{+}_{0,2}(\psi) = v^{=}_0(\psi) = (\mathsf{step}_{q_1})_\mathbb{M}(\mathtt{conf}_{q_\mathsf{init}, \mathtt{r}=0}, \mathtt{r}_{\geq 2}) = \mathtt{error}$. In the same way, we also have $\mu^{-}_{2,0}(\gamma) = \mathtt{error}$ with $\gamma = \mathsf{step}_{q_2}(\varphi_1, \mathsf{enc}(2))$. This reflects the fact that γ encodes the sequence resulting from appending $(q_2, 2)$ to the sequence encoded by φ_1, hence incrementing the value the counter while $\delta(q_1)$ required it to be decremented by one.

Finally, consider $\xi = \mathsf{step}_{q_3}(\varphi_2, \mathsf{enc}(1))$ encoding the sequence resulting from appending $(q_3, 1)$ to the sequence encoded by φ_2. As the value in the first counter was incremented, while the transition $\delta(q_2) = \langle 1^{\mathsf{test}}, q_3, q_3 \rangle$ required it to remain unchanged we obtain $\mu^{\neq}_{0,1}(\xi) = v^{=}_0(\xi) = \mathtt{error}$. △

Example 3.5. Consider the counter machine $\mathscr{C} = \langle 2, Q, q_\mathsf{init}, \delta \rangle$ with $Q = \{q_\mathsf{init}, q_1, q_2, q_3, q_4\}$ and δ as defined in following table

q	q_init	q_1	q_2	q_3	q_4
$\delta(q)$	$\langle 1^{++}, q_1 \rangle$	$\langle 2^{++}, q_2 \rangle$	$\langle 1^{\mathsf{test}}, q_4, q_3 \rangle$	$\langle 1^{++}, q_\mathsf{init} \rangle$	undefined

This machine does not have a finite computation, and its infinite computation loops indefinitely in the following cycle consisting of 4 transitions. It starts by incrementing both counters. Then it tests if the first counter has the value 0. As the counter has just been incremented, the test is bound to fail and hence that counter is decremented. It then increments the same counter, and returns to the initial state.

The halting state q_4 could only be reached if at a certain point the test would succeed, but this never happens since, at the point the tests are made, the value of the first counter is never 0. Thus, the machine $\mathscr{C}$ has the following infinite computation

$$\langle (q_\mathsf{init}, 0, 0), (q_1, 1, 0), (q_2, 1, 1), (q_3, 0, 1), (q_\mathsf{init}, 1, 1), (q_1, 2, 1), (q_2, 2, 2), (q_3, 1, 2), (q_\mathsf{init}, 2, 2), \ldots \rangle$$

As we show in the next section, since $\mathscr{C}$ has no finite computations, $\mathbb{M}_\mathscr{C}$ has no theorems. Let $k \geq 1$ and consider the sequence resulting from adding $(q_3, k-1, k+1)$ to the prefix of the infinite computation of $\mathscr{C}$ with $4(k-1)+3$ elements, and let φ_k encode this sequence. In this case, the value of the second counter is increased, when it should have remained the same, and we have

$$\mu^{\neq}_{k,k+1}(\varphi_k) = v_{k-1}(\varphi_k) = (\mathsf{step}_{q_3})_{\mathbb{M}_\mathscr{C}}(\mathtt{conf}_{q_2, \mathtt{r}_{\geq 1}, \mathtt{r}_{\geq 1}}, \mathtt{r}_{\geq 0}, \mathtt{r}_{\geq 2}) = \mathtt{error}. \qquad \triangle$$

4 Monadicity of Nmatrices is undecidable

In this section we show that $\mathbb{M}_\mathscr{C}$ really does what is intended. The main result of the paper then follows, after we additionally introduce a construction connecting the existence of a theorem with monadicity.

$\mathbb{M}_{\mathscr{C}}$ validates the finite computation of $\mathscr{C}$

In the following propositions we show that $\mathbb{M}_{\mathscr{C}}$ is interpreting computations of $\mathscr{C}$ as it should. Thus, formulas encoding computations of $\mathscr{C}$ that do not end in a halting state can be falsified, whilst the one encoding the finite computation of $\mathscr{C}$ is always designated.

Proposition 4.1. Let $\mathscr{C} = \langle n, Q, q_{\mathsf{init}}, \delta \rangle$ be a deterministic n-counter machine. If $\mathsf{nxt}(q, \vec{y}) = (q', \vec{z})$ then

$$(\mathsf{step}_{q'})_{\mathbb{M}_{\mathscr{C}}}(\mathtt{conf}_{q,\nu(\mathsf{enc}(\vec{y}))}, \nu(\mathsf{enc}(\vec{z}))) = \{\mathtt{conf}_{q',\nu(\mathsf{enc}(\vec{z}))}\}, \text{ for every } \nu \in \mathsf{Val}(\mathbb{M}_{\mathscr{C}}). \tag{1}$$

Proof. Suppose that $\mathsf{nxt}(q, \vec{y}) = (q', \vec{z})$. We have to consider three cases, depending on $\delta(q)$.

If $\delta(q) = (i^{++}, q')$ and $\vec{z} = \vec{y} + \vec{\mathsf{e}}_i$ then, for every $j \neq i$ and $\nu \in \mathsf{Val}(\mathbb{M}_{\mathscr{C}})$, we have $z_j = y_j$ and $\nu(\mathsf{enc}(z_j)) = \nu(\mathsf{enc}(y_j))$. Furthermore, $z_i = y_i + 1$, so $\mathsf{enc}(z_i) = \mathsf{suc}(\mathsf{enc}(y_i))$ and, for every $\nu \in \mathsf{Val}(\mathbb{M}_{\mathscr{C}})$, $\nu(\mathsf{enc}(z_i)) = \nu(\mathsf{suc}(\mathsf{enc}(y_i))) \in \mathsf{suc}_{\mathbb{M}_{\mathscr{C}}}(\nu(\mathsf{enc}(y_i)))$.

Otherwise, $\delta(q) = \langle i^{\mathsf{test}}, s_1, s_2 \rangle$ for some machine states s_1 and s_2.

If $s_1 = q'$, $y_i = 0$ and $\vec{z} = \vec{y}$. Then, for every $\nu \in \mathsf{Val}(\mathbb{M}_{\mathscr{C}})$, we have $\nu(\mathsf{enc}(\vec{z})) = \nu(\mathsf{enc}(\vec{y}))$ and $\nu(\mathsf{enc}(z_i)) = \nu(\mathsf{enc}(y_i)) = \nu(\mathsf{zero}) \in \{\mathtt{r}_{=0}, \mathtt{r}_{\geq 0}\}$.

If $s_2 = q'$, $y_i \neq 0$ and $\vec{y} = \vec{z} + \vec{\mathsf{e}}_i$. Then, for every $j \neq i$ and $\nu \in \mathsf{Val}(\mathbb{M}_{\mathscr{C}})$, we have $z_j = y_j$ and so $\nu(\mathsf{enc}(z_j)) = \nu(\mathsf{enc}(y_j))$. Furthermore, $y_i = z_i + 1$, so $\mathsf{enc}(y_i) = \mathsf{suc}(\mathsf{enc}(z_i))$ and, for every $\nu \in \mathsf{Val}(\mathbb{M}_{\mathscr{C}})$, $\nu(\mathsf{enc}(y_i)) = \nu(\mathsf{suc}(\mathsf{enc}(z_i))) \in \mathsf{suc}_{\mathbb{M}_{\mathscr{C}}}(\nu(\mathsf{enc}(z_i)))$.

In all the three cases we conclude (1) directly by the definition of $(\mathsf{step}_{q'})_{\mathbb{M}_{\mathscr{C}}}$. □

Theorem 4.2. Let $\mathscr{C} = \langle n, Q, q_{\mathsf{init}}, \delta \rangle$ be a deterministic n-counter machine with a finite computation $\langle C_0, \ldots, C_k \rangle$, then $\emptyset \vdash_{\mathbb{M}_{\mathscr{C}}} \mathsf{seq}(\langle C_0, \ldots, C_k \rangle)$.

Proof. Suppose that $\langle C_0, \ldots, C_k \rangle$ is the finite computation of $\mathscr{C}$. Then we have that $C_0 = (q_{\mathsf{init}}, \vec{\mathsf{zero}})$, where $\vec{\mathsf{zero}} = \langle \mathsf{zero}, \ldots, \mathsf{zero} \rangle$, $\mathsf{nxt}(C_j) = C_{j+1}$ for every $0 \leq j < k$ and $C_k = (q_k, \vec{z})$ is a halting configuration. For every $\nu \in \mathsf{Val}(\mathbb{M}_{\mathscr{C}})$ we have $\nu(\mathsf{seq}(\langle C_0 \rangle)) = \mathtt{conf}_{q_{\mathsf{init}},\nu(\vec{\mathsf{zero}})}$ and, by proposition 4.1, $\nu(\langle C_0, \ldots, C_k \rangle) = \mathtt{conf}_{q_k,\nu(\mathsf{enc}(\vec{z}))}$, where $C_k = (q_k, \vec{z})$. Since C_k is a halting configuration, we conclude that $\mathtt{conf}_{q_k,\nu(\mathsf{enc}(\vec{z}))} \in D$, and so $\emptyset \vdash_{\mathbb{M}_{\mathscr{C}}} \mathsf{seq}(\langle C_0, \ldots, C_k \rangle)$. □

$\mathbb{M}_{\mathscr{C}}$ can falsify everything else

The following propositions deal with all the possible ways in which a formula can fail to represent a halting computation of $\mathscr{C}$.

Proposition 4.3. Let $\mathscr{C} = \langle n, Q, q_{\mathsf{init}}, \delta \rangle$ be a deterministic n-counter machine. If $\varphi \in \mathbf{L}_{\Sigma_{\mathscr{C}}}(\emptyset)$ does not represent a sequence of configurations of $\mathscr{C}$ then $\nu(\varphi) \neq \mathtt{conf}_{q,\vec{y}}$ for all $\nu \in \mathsf{Val}_{\mathscr{C}}$, $q \in Q$ and $\vec{y} \in \mathbf{Rm}^n$.

Proof. The proof follows by induction on the structure of the formula $\varphi \in \mathbf{L}_{\Sigma_{\mathscr{C}}}(\emptyset)$.

In the base case we have $\varphi \in \{\mathsf{zero}, \varepsilon\}$. The statement then holds since, for all $\nu \in \mathsf{Val}_{\mathscr{C}}$, $\nu(\varphi) \in \{\mathsf{zero}, \mathtt{init}\}$.

For the step we have two cases. In the first case, $\varphi = \mathsf{suc}(\psi)$ and $\nu(\mathsf{suc}(\psi)) \in \mathbf{Rm} \cup \{\mathtt{error}\}$. In the second case, $\varphi = \mathtt{step}_q(\psi, \psi_1, \ldots, \psi_n)$ and, if φ does not represent any sequence of configurations, then one of the following must hold

- ψ does not represent a sequence of configurations, in which case, by induction hypothesis, $\nu(\psi) \neq \mathtt{conf}_{q,\vec{y}}$, or
- $\psi_i \neq \mathsf{enc}(j)$, for some $1 \leq i \leq n$, so $\nu(\psi_i) \notin \mathbf{Rm}$.

In both cases we have $v(\varphi) = \mathsf{suc}_{\mathbb{M}_{\mathscr{C}}}(v(\psi), v(\psi_1), \ldots, v(\psi_n)) = \mathtt{error}$. □

Proposition 4.4. Given a deterministic counter machine $\mathscr{C} = \langle n, Q, q_{\mathsf{init}}, \delta \rangle$. If $\mathsf{nxt}(q, \vec{y}) \neq (q', \vec{z})$ then

$$(\mathsf{step}_{q'})_{\mathbb{M}_{\mathscr{C}}}(\mathtt{conf}_{q, v(\mathsf{enc}(\vec{y}))}, v(\mathsf{enc}(\vec{z}))) = \{\mathtt{error}\}, \tag{2}$$

for some $v \in \mathsf{Val}(\mathbb{M}_{\mathscr{C}})$.

Proof. Assume $\mathsf{nxt}(q, \vec{y}) \neq (q', \vec{z})$ and notice that, if $\delta(q)$ concerns the ith counter and $y_j \neq z_j$, for some $j \neq i$, then $\mu^{\neq}_{y_j, z_j}(\mathsf{enc}(y_j)) \neq \mu^{\neq}_{y_j, z_j}(\mathsf{enc}(z_j))$, by remark 3.3. Therefore, equality (2) holds for $v = \mu^{\neq}_{y_j, z_j}$. Because of this, throughout the rest of the proof, we assume that, $y_j = z_j$, for every $j \neq i$, and concern ourselves only with the values taken by y_i and z_i. We have to consider three cases, depending on $\delta(q)$.

If $\delta(q) = \langle i^{++}, s \rangle$, for some machine state s. Then either $q' \neq s$, in which case equality (2) holds for every $v \in \mathsf{Val}(\mathbb{M}_{\mathscr{C}})$, or $z_i \neq y_i + 1$. In this later case, also by remark 3.3, we have $\mu^{+}_{y_i, z_i}(\mathsf{enc}(z_i)) \notin \mathsf{suc}_{\mathbb{M}_{\mathscr{C}}}(\mu^{+}_{y_i, z_i}(\mathsf{enc}(y_i)))$, so equality (2) holds for $v = \mu^{+}_{y_i, z_i}$.

Otherwise, $\delta(q) = \langle i^{\mathsf{test}}, s_1, s_2 \rangle$ for some machine states s_1 and s_2.

If $y_i = 0$, then either $q' \neq s_1$ or $z_i \neq y_i$. Consider the valuation $\mu^{=}_{y_i, z_i} = v^{=}_0$. Since $v^{=}_0(y_i) = \mathtt{r}_{=0} \notin \mathsf{suc}_{\mathbb{M}_{\mathscr{C}}}(v^{=}_0(\mathsf{enc}(z_i)))$, the second condition concerning i^{test}, in the definition of $(\mathsf{step}_{q'})_{\mathbb{M}_{\mathscr{C}}}$, is not satisfied whenever $v = \mu^{=}_{y_i, z_i}$. The first condition is also not satisfied if $q' \neq s_1$, directly, or if $z_i \neq y_i$, since in this case $\mu^{=}_{y_i, z_i}(z_i) \neq \mu^{=}_{y_i, z_i}(y_i)$, by remark 3.3. We conclude that equality (2) holds for $v = \mu^{=}_{y_i, z_i}$.

If $y_i \neq 0$, then either $q' \neq s_2$ or $z_i \neq y_i - 1$. Note that, in any case, $\mu^{-}_{y_i, z_i}(\mathsf{enc}(y_i)) \notin \{\mathtt{r}_{=0}, \mathtt{r}_{\geq 0}\}$, which can easily be checked using the definition of $\mu^{-}_{y_i, z_i}$. Therefore, the first condition concerning i^{test}, in the definition of $(\mathsf{step}_{q'})_{\mathbb{M}_{\mathscr{C}}}$, is not satisfied whenever $v = \mu^{-}_{y_i, z_i}$. The second condition is also not satisfied if $q' \neq s_2$, directly, or if $z_i \neq y_i - 1$, since in this case $\mu^{-}_{y_i, z_i}(\mathsf{enc}(y_i)) \notin \mathsf{suc}_{\mathbb{M}_{\mathscr{C}}}(\mu^{-}_{y_i, z_i}(\mathsf{enc}(z_i)))$, by remark 3.3. We conclude that equality (2) holds for $v = \mu^{-}_{y_i, z_i}$. □

Proposition 4.5. Given a deterministic counter machine $\mathscr{C} = \langle n, Q, q_{\mathsf{init}}, \delta \rangle$ and $\varphi \in \mathbf{L}_{\Sigma_{\mathscr{C}}}(\emptyset)$ such that $\varphi = \mathsf{seq}(\langle C_0, \ldots, C_k \rangle)$. If $\langle C_0, \ldots, C_k \rangle$ is not the computation of $\mathscr{C}$ then $\emptyset \not\models_{\mathbb{M}_{\mathscr{C}}} \varphi$.

Proof. If $\langle C_0, \ldots, C_k \rangle$ is not the computation of $\mathscr{C}$, then one of the following must hold: (i) C_0 is not the initial configuration of $\mathscr{C}$, (ii) C_k is not a halting configuration of $\mathscr{C}$, or (iii) there is some $1 \leq i < k$ such that $\mathsf{nxt}(C_i) \neq C_{i+1}$. We deal with each situation separately.

If (i) holds and $C_0 = (q, \vec{y})$ then either $q \neq q_{\mathsf{init}}$ or $y_j \neq 0$, for some $1 \leq j \leq n$. In the first case, for all $v \in \mathsf{Val}(\mathbb{M}_{\mathscr{C}})$, we have $(\mathsf{step}_q)_{\mathbb{M}_{\mathscr{C}}}(\mathtt{init}, v(\mathsf{enc}(\vec{y}))) = \mathtt{error}$. In the second case, $v_{y_j - 1}(\mathsf{enc}(y_j)) \notin \{\mathtt{r}_{=0}, \mathtt{r}_{\geq 0}\}$ and $(\mathsf{step}_{q_{\mathsf{init}}})_{\mathbb{M}_{\mathscr{C}}}(\mathtt{init}, v_{y_j - 1}(\mathsf{enc}(\vec{y}))) = \mathtt{error}$.

If (ii) holds and $C_k = (q, \vec{y})$ then q is not a halting state and $v(\varphi) \in \{\mathtt{error}, \mathtt{conf}_{q, v(\vec{y})}\} \subseteq A \setminus D$, for all $v \in \mathsf{Val}(\mathbb{M}_{\mathscr{C}})$.

If (iii) holds then, by proposition 4.4, there is $v \in \mathsf{Val}(\mathbb{M}_{\mathscr{C}})$ such that $v(\mathsf{seq}(\langle C_0, \ldots, C_{i+1} \rangle)) = \mathtt{error}$.

In any of the cases, there is some $v \in \mathsf{Val}(\mathbb{M}_{\mathscr{C}})$ such that $v(\varphi) \notin D$, so $\emptyset \not\models_{\mathbb{M}_{\mathscr{C}}} \varphi$. □

Having seen how to refute any formula not representing a computation of $\mathscr{C}$ we conclude $\mathbb{M}_{\mathscr{C}}$ does exactly what we intended.

Theorem 4.6. Let $\mathscr{C} = \langle n, Q, q_{\mathsf{init}}, \delta \rangle$ be a deterministic counter machine. For any formula $\varphi \in \mathbf{L}_{\Sigma_{\mathscr{C}}}(P)$ we have $\emptyset \vdash_{\mathbb{M}_{\mathscr{C}}} \varphi$ if and only if $\varphi = \mathsf{seq}(\langle C_0, \ldots, C_k \rangle)$ and $\langle C_0, \ldots, C_k \rangle$ is a finite computation of $\mathscr{C}$.

Proof. From right to left, if $\langle C_0, \ldots, C_k \rangle$ is a finite computation of $\mathscr{C}$ and $\varphi = \mathsf{seq}(\langle C_0, \ldots, C_k \rangle)$ then, by theorem 4.2, we have that $\emptyset \vdash_{\mathbb{M}_{\mathscr{C}}} \varphi$. In the other direction, suppose $\emptyset \vdash_{\mathbb{M}_{\mathscr{C}}} \varphi$ then, as discussed in example 3.2, φ must be a closed formula. By proposition 4.3, $\varphi = \mathsf{seq}(\langle C_0, \ldots, C_k \rangle)$ for some sequence

of configurations $\mathsf{seq}(\langle C_0,\ldots,C_k\rangle)$, and, by proposition 4.5, $\mathsf{seq}(\langle C_0,\ldots,C_k\rangle)$ must be the computation of $\mathscr{C}$. □

From theoremhood to monadicity

In order to obtain the announced undecidability result we need one last construction. We will show how to build an Nmatrix $\mathbb{M}_\mathsf{m}$ from an Nmatrix $\mathbb{M}$, under certain conditions, such that $\mathbb{M}_\mathsf{m}$ is monadic if and only if $\vdash_\mathbb{M}$ has theorems.

Given a finite Σ-Nmatrix $\mathbb{M}=\langle A,\cdot_\mathbb{M},D\rangle$, let Σ_m be such that $\Sigma^{(2)}_\mathsf{m}=\Sigma^{(2)}\cup\{f_a : a\in A\}$ and $\Sigma^{(k)}_\mathsf{m}=\Sigma^{(k)}$, for every $k\neq 2$.

Let $A_\mathsf{m}=A\cup\{1\}$, assuming w.l.g. that $1\notin A$, consider $\mathbb{M}_\mathsf{m}=\langle A_\mathsf{m},\cdot_\mathsf{m},\{1\}\rangle$ the Σ_m-Nmatrix where, for each $g\in\Sigma^{(k)}$,

$$g_\mathsf{m}(x_1,\ldots,x_k)=\begin{cases} g_\mathbb{M}(x_1,\ldots,x_k) & \text{if } x_1,\ldots,x_k\in A\\ A_\mathsf{m} & \text{otherwise}\end{cases}$$

and, for every $a\in A$,

$$(f_a)_\mathsf{m}(x,y)=\begin{cases}\{1\} & \text{if } x=a \text{ and } y\in D\\ A & \text{if } x\in A\setminus\{a\} \text{ and } y\in D\\ A_\mathsf{m} & \text{otherwise}\end{cases}$$

The following theorem targets Nmatrices with infectious values. Recall that $*$ is infectious in $\mathbb{M}$ if for every connective © in the signature of $\mathbb{M}$ we have $©_\mathbb{M}(x_1,\ldots,x_k)=*$ whenever $*\in\{x_1,\ldots,x_k\}$.

Proposition 4.7. Given Nmatrix $\mathbb{M}$ with at least two truth-values and among them an infectious non-designated value, $\vdash_\mathbb{M}$ has theorems if and only if $\mathbb{M}_\mathsf{m}$ is monadic.

Proof. Let us denote the infectious non-designated value of $\mathbb{M}$ by $*$. Clearly, $*$ ceases to be infectious in $\mathbb{M}_\mathsf{m}$ as $(f_a)_{\mathbb{M}_\mathsf{m}}$ does not necessarily output $*$ when it receives it as input. The value 1 is also not infectious in $\mathbb{M}_\mathsf{m}$, quite the opposite, when given as input to any connective the output can take any value. That is, for every connective $©\in\Sigma_\mathsf{m}$ we have $©_{\mathbb{M}_\mathsf{m}}(x_1,\ldots,x_k)=A_\mathsf{m}$ whenever $1\in\{x_1,\ldots,x_k\}$. This immediately implies that if $\psi\in\mathsf{Sub}(\varphi)\setminus\{\varphi\}$ and $1\in\psi_{\mathbb{M}_\mathsf{m}}(x)$ then $\varphi_{\mathbb{M}_\mathsf{m}}(x)=A_\mathsf{m}$ for any $x\in A_\mathsf{m}$.

If $\emptyset\vdash_\mathbb{M}\varphi$ then φ must be a closed formula due to the presence of $*$. Hence, for $v\in\mathsf{Val}(\mathbb{M}_\mathsf{m})$ we have $v(\varphi)\in D$. Thus, $\{p\}\cup\{f_a(p,\varphi) : a\in A\}$ is a set of monadic separators for $\mathbb{M}_\mathsf{m}$, as p separates 1 from the elements in A, and $f_a(p,\varphi)$ separates a from every $b\in A$.

If instead there are no theorems in $\vdash_\mathbb{M}$, let us consider an arbitrary monadic formula $\varphi\in L_{\Sigma_\mathsf{m}}(\{p\})$ and show it cannot separate $*$ from the other elements of A. We need to consider two cases.

- If $\varphi\in L_\Sigma(\{p\})$ then $\varphi_{\mathbb{M}_\mathsf{m}}(a)=\varphi_\mathbb{M}(a)\subseteq A\not\ni 1$ for every $a\in A$. In which case φ cannot separate any pair of distinct elements of A and, in particular, cannot separate $*$ from any other element of A.
- If $\varphi\in L_{\Sigma_\mathsf{m}}(\{p\})\setminus L_\Sigma(\{p\})$ then there is $f_a(\psi_1,\psi_2)\in\mathsf{Sub}(\varphi)$ with $\psi_1,\psi_2\in L_\Sigma(\{p\})$. If p occurs in ψ_2 then $(\psi_2)_{\mathbb{M}_\mathsf{m}}(*)=(\psi_2)_\mathbb{M}(*)=\{*\}$ and $(f_a(\psi_1,\psi_2))_{\mathbb{M}_\mathsf{m}}(*)=A_\mathsf{m}$, since $*\notin D$. If p does not occur in ψ_2 then, since $\emptyset\not\vdash_\mathbb{M}\psi_2$, $(\psi_2)_\mathbb{M}\cap(A\setminus D)\neq\emptyset$ and we also obtain $(f_a(\psi_1,\psi_2))_{\mathbb{M}_\mathsf{m}}(*)=A_\mathsf{m}$. Therefore, $\varphi_{\mathbb{M}_\mathsf{m}}(*)=A_\mathsf{m}$ since either $\varphi=f_a(\psi_1,\psi_2)$ or $f_a(\psi_1,\psi_2)\in\mathsf{Sub}(\varphi)\setminus\{\varphi\}$ and $1\in(f_a(\psi_1,\psi_2))_{\mathbb{M}_\mathsf{m}}(*)$. As $\varphi_{\mathbb{M}_\mathsf{m}}(*)$ contains both designated and non-designated elements it cannot separate $*$ from any other element of A.

As we are assuming that A has at least two elements, we conclude that $\mathbb{M}_\mathsf{m}$ is not monadic. □

Finally, we get to the main result of this paper.

Theorem 4.8. The problem of determining if a given finite Σ-Nmatrix is monadic is undecidable.

Proof. For every counter machine $\mathscr{C}$, the Nmatrix $\mathbb{M}_{\mathscr{C}}$ is in the conditions of Theorem 4.7, as it has more than two truth-values and `error` is infectious. Therefore, by applying successively Theorem 4.6 and 4.7, we reduce the halting problem for counter machines to the problem of checking if a finite Nmatrix is monadic. Indeed, for a given counter machine $\mathscr{C}$, $\mathscr{C}$ halts if and only if $\vdash_{\mathbb{M}_{\mathscr{C}}}$ has theorems if and only if $(\mathbb{M}_{\mathscr{C}})_{\mathsf{m}}$ is monadic. Furthermore, the presented constructions are all computable and $(\mathbb{M}_{\mathscr{C}})_{\mathsf{m}}$ is always finite since, if $\mathscr{C}$ has m states and n counters, then $\mathbb{M}_{\mathscr{C}}$ has $m \times 4^n + 6$ truth-values and $\Sigma_{\mathscr{C}}$ has $3 + m$ connectives. Therefore, $(\mathbb{M}_{\mathscr{C}})_{\mathsf{m}}$ has $m \times 4^n + 7$ truth-values and $(\Sigma_{\mathscr{C}})_{\mathsf{m}}$ has $m \times 4^n + m + 9$ connectives. We can therefore conclude the proof just by invoking Theorem 3.1. □

As a simple corollary we obtain that following result about Nmatrices, or better, about their underlying multi-algebras.

Corollary 4.9. The problem of generating all expressible unary multi-functions in an arbitrary finite Nmatrix is not computable.

Proof. Just note that if we could compute the set of all expressible unary multi-functions, as the set is necessarily finite, we could test each of them for the separation of values, as illustrated in the case of matrices in Example 2.3. □

5 Conclusion

In this paper we have shown that, contrarily to the most common case of logical matrices, the monadicity property is undecidable for non-deterministic matrices. As a consequence, we conclude that the set of all multi-functions expressible in a given finite Nmatrix is not computable, in general. These results, of course, do not spoil the usefulness of the techniques for obtaining axiomatizations, analytical calculi and automated proof-search for monadic non-deterministic matrices. This is especially the case since, for a given Nmatrix, one can always define a monadic Nmatrix over an enriched signature, such that its logic is a conservative extension of the logic of the previous Nmatrix, as described in [14]. The results show, however, that tool support for logics based on non-deterministic matrices must necessarily have its limitations.

On a closer perspective, the reduction we have obtained from counter machines to Nmatrices (of which non-determinism is a fundamental ingredient) just adds to the initial perception that allowing for non-determinism brings a substantial amount of expressive power to logical matrices. Concretely, it opens the door for studying the computational hardness of other fundamental meta-theoretical questions regarding logics defined by Nmatrices. In particular, we will be interested in studying the problem of deciding whether two given finite Nmatrices define the same logic, a fundamental question raised by Zohar and Avron in [5], for which only necessary or sufficient conditions are known.

Additionally, we deem it important to further explore the connections between Nmatrices and term-dag-automata (an interesting computational model for term languages [10, 1]) and which informed our undecidability result. Another relevant direction for further investigation is the systematic study of infectious semantics, in the lines of [8, 6], whose variable inclusion properties also played an important role in our results.

References

[1] Siva Anantharaman, Paliath Narendran & Michael Rusinowitch (2005): *Closure properties and decision problems of dag automata*. Information Processing Letters 94(5), pp. 231–240, doi:10.1016/j.ipl.2005.02.004.

[2] Arnon Avron, Beata Konikowska & Anna Zamansky (2013): *Cut-free Sequent Calculi for C-systems with Generalized Finite-valued Semantics*. Journal of Logic and Computation 23(3), pp. 517–540, doi:10.1093/logcom/exs039.

[3] Arnon Avron & Iddo Lev (2005): *Non-deterministic multiple-valued structures*. Journal of Logic and Computation 15(3), pp. 241–261, doi:10.1093/logcom/exi001.

[4] Arnon Avron & Anna Zamansky (2011): *Non-deterministic semantics for logical systems*. In Dov M. Gabbay & Franz Guenthner, editors: Handbook of Philosophical Logic, 16, Springer, pp. 227–304, doi:10.1007/978-94-007-0479-4_4.

[5] Arnon Avron & Yoni Zohar (2019): *Rexpansions of non-deterministic matrices and their applications*. The Review of Symbolic Logic 12(1), pp. 173–200, doi:10.1017/S1755020318000321.

[6] Stefano Bonzio, Tommaso Moraschini & Michele Pra Baldi (2020): *Logics of left variable inclusion and Płonka sums of matrices*. Archive for Mathematical Logic 60, pp. 49–76, doi:10.1007/s00153-020-00727-6.

[7] Carlos Caleiro & Sérgio Marcelino (2019): *Analytic calculi for monadic PNmatrices*. In Rosalie Iemhoff, Michael Moortgat & Ruy de Queiroz, editors: Logic, Language, Information, and Computation (WoLLIC 2019), LNCS 11541, Springer-Verlag, pp. 84–98, doi:10.1007/978-3-662-59533-6_6.

[8] Carlos Caleiro, Sérgio Marcelino & Pedro Filipe (2020): *Infectious semantics and analytic calculi for even more inclusion logics*. In: 2020 IEEE 50th International Symposium on Multiple-Valued Logic, pp. 224–229, doi:10.1109/ISMVL49045.2020.000-1.

[9] Carlos Caleiro, João Marcos & Marco Volpe (2015): *Bivalent semantics, generalized compositionality and analytic classic-like tableaux for finite-valued logics*. Theoretical Computer Science 603, pp. 84–110, doi:10.1016/j.tcs.2015.07.016.

[10] Witold Charatonik (1999): *Automata on DAG Representations of Finite Trees*. Technical Report MPI-I-1999-2-001, Max-Planck-Institut für Informatik, Saarbrücken, Germany.

[11] Josep Font (2016): *Abstract Algebraic Logic*. Mathematical Logic and Foundations 60, College Publications.

[12] Dietlinde Lau (2006): *Function Algebras on Finite Sets, A Basic Course on Many-Valued Logic and Clone Theory*. Springer Monographs in Mathematics, Springer, doi:10.1007/3-540-36023-9.

[13] Sérgio Marcelino & Carlos Caleiro (2017): *Disjoint fibring of non-deterministic matrices*. In Juliette Kennedy & Ruy J.G.B. de Queiroz, editors: Logic, Language, Information, and Computation (WoLLIC 2017), LNCS 10388, Springer-Verlag, pp. 242–255, doi:10.1007/978-3-662-55386-2_17.

[14] Sérgio Marcelino & Carlos Caleiro (2019): *Axiomatizing non-deterministic many-valued generalized consequence relations*. Synthese 198, pp. 5373–5390, doi:10.1007/s11229-019-02142-8.

[15] Marvin Minsky (1967): *Computation: Finite and Infinite Machines*. Prentice-Hall.

[16] David John Shoesmith & Timothy Smiley (1978): *Multiple-conclusion logic*. Cambridge University Press, doi:10.1017/CBO9780511565687.

[17] Ryszard Wójcicki (1988): *Theory of Logical Calculi*. Synthese Library 199, Kluwer, doi:10.1007/978-94-015-6942-2.