Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 351 (2020) 167–186

www.elsevier.com/locate/entcs

An Efficient Algorithm for Representing Piecewise Linear Functions into Logic

Sandro Preto[1,3] Marcelo Finger[1,2,4]

Institute of Mathematics and Statistics
University of São Paulo
São Paulo, Brazil

Abstract

Rational McNaughton functions may be implicitly represented by logical formulas in Łukasiewicz Infinitely-valued Logic by constraining the set of valuations to the ones that satisfy some specific formulas. This work investigates this implicit representation called representation modulo satisfiability and describes a polynomial algorithm that builds it — the representative formula and the constraining ones — for a given rational McNaughton function.

Keywords: Łukasiewicz Infinitely-valued Logic, Rational McNaughton Functions, Piecewise Linear Functions.

1 Introduction

The ability to represent any piecewise linear function with a logical formula allows us to apply automated reasoning techniques to the study of real systems, whose behavior is either modeled or approximated by such a function. However such an ability will only be effective if there are efficient ways to generate a formula in a target logic in which reasoning is not exceedingly complex. Classical logic with its binary semantics, despite of being a natural target for representing Boolean functions, may not be the natural way to represent continuous functions which inevitably would require some encoding of rational or real numbers; so we follow the path of electing some form of many-valued logic, whose semantics range over rational numbers, as a more adequate representation framework. That path has initially been explored by applications in fuzzy control [5].

[1] This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

[2] Partly supported by Fapesp projects 2014/12236-1 and 2019/07665-4 and CNPq grant PQ 303609/2018-4.

[3] Email: spreto@ime.usp.br

[4] Email: mfinger@ime.usp.br

https://doi.org/10.1016/j.entcs.2020.08.009

Neural network interpretability is a challenge to the development of artificial intelligence and is also another motivation for the representation of piecewise linear functions, as described by [4]. In fact, a neural network, depending on its class of activation functions, can be seen either as a piecewise linear function or as a continuous function that can be approximated by one [12].

The first candidate to consider as a target logic is Łukasiewicz Infinitely-valued Logic ($Ł_\infty$), arguably one of the best studied many-valued logics [7]; it has several interesting properties such as continuous truth-functional semantics, classical logic as limit case, and well developed proof-theoretical and algebraic presentations. Its formulas are known to represent exactly the so-called McNaughton functions, consisting of $[0,1]$-valued piecewise linear functions *with integer coefficients* over $[0,1]^n$ [13,15]. This restriction to integer coefficients fails to fulfill, for instance, the hypotheses for the classic Stone-Weierstrass Approximation Theorem [16] and there is no known analogous to Proposition 1.1 below for McNaughton functions.

This issue is circumvented by slightly modifying McNaughton functions to allow their linear pieces to have rational coefficients — the rational McNaughton functions; such generalization is enough to perform Weierstrass-like approximations [5,2]. Figure 1 shows a continuous function $f : [0,1] \to [0,1]$ (a) with two possible approximations by rational McNaughton functions: with two (b) and five different linear pieces (c). Note that continuous functions with more general domain and range might be normalized in order to perform such approximations.

Proposition 1.1 (Variation of Weierstrass Approximation Theorem [5]) *Let $f : [0,1]^n \to [0,1]$ be a continuous function and $\varepsilon > 0$. Then there is a rational McNaughton function $\tilde{f} : [0,1]^n \to [0,1]$ such that $|f(\mathbf{x}) - \tilde{f}(\mathbf{x})| < \varepsilon$, for all $\mathbf{x} \in [0,1]^n$.*

In this context, the target logic must be a system, preferably based on $Ł_\infty$, with semantics that comprehends all rational McNaughton functions and, given one such function, be endowed with an efficient algorithm that provides a formula that represents it. Furthermore, we highlight that it would be of little practical use if the reasoning complexity in such system is exceedingly high.

Esteva, Godo & Montagna propose logic $ŁΠ\frac{1}{2}$ which extends $Ł_\infty$ with a product operator, its residuum, and a constant expressing the truth value $\frac{1}{2}$, not directly expressible in $Ł_\infty$ [8]. That logic not only allows for the expressivity of rational McNaughton functions but also expresses piecewise polynomials; as a consequence satisfiability over $ŁΠ\frac{1}{2}$ requires finding roots of polynomials of n-degree rendering its complexity extremely high. Aguzzoli & Mundici propose logic $\exists Ł$ which also expresses rational McNaughton functions and has complexity Σ_2^p for the satisfiability problem [2,3]. Logic $\exists Ł$ extends $Ł_\infty$ and introduces rational numbers by providing restricted form of propositional quantification whose semantic counterpart is the maximization of a set of $Ł_\infty$-valuations of a formula.

Gerla introduces Rational Łukasiewicz Logic by extending $Ł_\infty$ with division operators δ_n that induces division by $n \in \mathbb{N}^*$ in its semantics [11]; its associated tautology problem is coNP-complete, which is a reasonable complexity for this task.

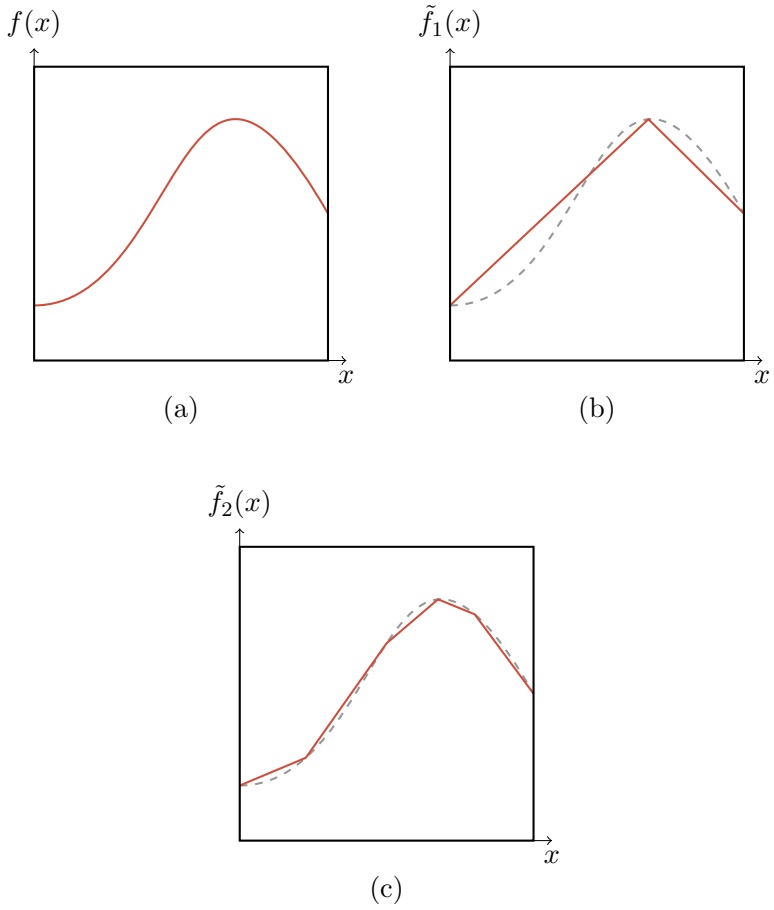

Fig. 1. Continuous one-variable function approximated by rational McNaughton functions.

This logic expresses all rational McNaughton functions however it was not provided an algorithm to build the representative formulas, and an attempt to derive one from the results in [11] would lead to the problem of representing McNaughton functions in $Ł_\infty$; it is known that this task may be done in polynomial time on the coefficients of some specific functions [1], however these methods lead to exponential time complexity if binary representation of the coefficients is used.

Finger & Preto provide a way to *implicitly* express rational McNaughton functions in $Ł_\infty$ called representation modulo satisfiability ($Ł_\infty$-MODSAT) [10]. For that, in addition to a representative formula, it is introduced a set of formulas that constrains $Ł_\infty$-valuations to those that satisfy all formulas in the set; a rational McNaughton function f is then represented by a pair $\langle \varphi, \Phi \rangle$, where φ is a formula that semantically acquires values $f(\mathbf{x})$, for $\mathbf{x} \in [0,1]^n$, from valuations in $\{v(\psi) = 1 \mid \psi \in \Phi\}$, where Φ is a set of formulas. Instead of an extension of the logic, this proposal works in $Ł_\infty$ itself, which has computational problems with reasonable complexity — e.g., satisfiability over $Ł_\infty$ is NP-complete [14]. Also, there

already are available implementations of $Ł_\infty$-solvers which are discussed in the literature and for which phase transition phenomenon is identified [6,9]. Unfortunately, an attempt to derive a representation builder algorithm from results in [10] would also lead to an exponential blow up, since the proposed pairs for representing only truncated linear functions are already exponential in the binary representation of their coefficients.

Our goal here is to provide an efficient algorithm that, given a rational McNaughton function, outputs a pair $\langle \varphi, \Phi \rangle$ that represents it in the target system $Ł_\infty$-MODSAT. We show that all rational McNaughton functions may be represented modulo satisfiability by a constructive proof from which we derive a polynomial algorithm that builds such representation.

This paper is organized as follows: Section 2 introduces all necessary concepts of Łukasiewicz Infinitely-valued Logic and the definition of rational McNaughton functions; Section 3 has the formalization of the concept of representation modulo satisfiability; Section 4 provides a convention on rational McNaughton functions encoding for computation purposes as well as some results about these functions; Section 5 has a theoretical and algorithmic treatment of a particular case of representation modulo satisfiability of rational McNaughton functions which are truncated linear functions; and Section 6 finally treats, also theoretically and algorithmically, the representation modulo satisfiability of general rational McNaughton functions.

2 Preliminaries

The basic language $\mathcal{L}$ of Łukasiewicz Infinitely-valued Logic ($Ł_\infty$) comprehends the formulas built from a countable set of propositional variables $\mathbb{P}$, and disjunction ($\oplus$) and negation ($\neg$) operators. For the semantics, define a valuation as a function $v : \mathcal{L} \to [0, 1]$, such that, for $\varphi, \psi \in \mathcal{L}$:

$$v(\varphi \oplus \psi) = \min(1, v(\varphi) + v(\psi)); \tag{1}$$
$$v(\neg\varphi) = 1 - v(\varphi). \tag{2}$$

One may just give a function $v_\mathbb{P}$ which maps propositional variables to a value in the interval $[0, 1]$ and extend this function to a valuation by obeying (1) and (2). This extension is uniquely defined by such assignment to the variables in $\mathbb{P}$ given by $v_\mathbb{P}$.

We denote by **Val** the set of all valuations; by $\mathrm{Var}(\Phi)$ the set of all propositional variables occurring in the formulas $\varphi \in \Phi$; and by $\mathbf{X}_n$ the set of propositional variables $\{X_1, \ldots, X_n\} \subset \mathbb{P}$. A formula φ is *satisfiable* if there exists a $v \in \mathbf{Val}$ such that $v(\varphi) = 1$; otherwise it is *unsatisfiable*. A set of formulas Φ is satisfiable if there exists a $v \in \mathbf{Val}$ such that $v(\varphi) = 1$, for all $\varphi \in \Phi$. We denote by $\mathbf{Val}_\Phi$ the set of all valuations $v \in \mathbf{Val}$ that satisfies a set of formulas Φ.

From disjunction and negation we derive the following operators:

Conjunction:	$\varphi \odot \psi =_{\text{def}} \neg(\neg\varphi \oplus \neg\psi)$	$v(\varphi \odot \psi) = \max(0, v(\varphi) + v(\psi) - 1)$
Implication:	$\varphi \to \psi =_{\text{def}} \neg\varphi \oplus \psi$	$v(\varphi \to \psi) = \min(1, 1 - v(\varphi) + v(\psi))$
Maximum:	$\varphi \vee \psi =_{\text{def}} \neg(\neg\varphi \oplus \psi) \oplus \psi$	$v(\varphi \vee \psi) = \max(v(\varphi), v(\psi))$
Minimum:	$\varphi \wedge \psi =_{\text{def}} \neg(\neg\varphi \vee \neg\psi)$	$v(\varphi \wedge \psi) = \min(v(\varphi), v(\psi))$
Bi-implication:	$\varphi \leftrightarrow \psi =_{\text{def}} (\varphi \to \psi) \wedge (\psi \to \varphi)$	$v(\varphi \leftrightarrow \psi) = 1 - \lvert v(\varphi) - v(\psi)\rvert$

Note that $v(\varphi \to \psi) = 1$ iff $v(\varphi) \leq v(\psi)$; similarly, $v(\varphi \leftrightarrow \psi) = 1$ iff $v(\varphi) = v(\psi)$. Let X be a propositional variable, then, $v(X \odot \neg X) = 0$, for any $v \in \mathbf{Val}$; we define the constant $\mathbf{0}$ by $X \odot \neg X$. We also define $0\varphi =_{\text{def}} \mathbf{0}$ and $n\varphi =_{\text{def}} \varphi \oplus \cdots \oplus \varphi$, n times, for $n \in \mathbb{N}^*$; and $\bigoplus_{i \in \varnothing} \varphi_i =_{\text{def}} \mathbf{0}$.

Adapting the definition in [7], a *rational McNaughton function* $f : [0,1]^n \to [0,1]$ is a function that satisfies the following conditions:

- f is continuous with respect to the usual topology of $[0,1]$ as an interval of the real number line;
- There are linear polynomials $p_1, \ldots, p_m$ over $[0,1]^n$ with rational coefficients such that, for each point $\mathbf{x} \in [0,1]^n$, there is an index $i \in \{1, \ldots, m\}$ with $f(\mathbf{x}) = p_i(\mathbf{x})$. Polynomials $p_1, \ldots, p_m$ are the *linear pieces* of f.

3 Representation Modulo Φ-Satisfiable

A *McNaughton function* is a rational McNaughton function whose linear pieces have integer coefficients. Let φ be a $Ł_\infty$-formula with $\text{Var}(\varphi) \subset \mathbf{X}_n$, we inductively associate to φ a function $f_\varphi : [0,1]^n \to [0,1]$ by:

(i) $f_{X_j}(x_1, \ldots, x_n) = x_j$, for $j = 1, \ldots, n$;

(ii) $f_{\neg\varphi}(x_1, \ldots, x_n) = 1 - f_\varphi(x_1, \ldots, x_n)$;

(iii) $f_{\varphi_1 \oplus \varphi_2}(x_1, \ldots, x_n) = \min(1, f_{\varphi_1}(x_1, \ldots, x_n) + f_{\varphi_2}(x_1, \ldots, x_n))$.

We have that f_φ is a McNaughton function such that

$$f_\varphi(v(X_1), \ldots, v(X_n)) = v(\varphi), \text{ for } v \in \mathbf{Val}. \tag{3}$$

Reciprocally, McNaughton's Theorem [13] states that, for any McNaughton function f, there is a formula φ such that $f = f_\varphi$. We say that φ *represents* f.

Although formulas of $Ł_\infty$ only represent (integer) McNaughton functions, we take the strategy of restricting the set $\mathbf{Val}$ of valuations in order to implicitly represent rational McNaughton functions. For that, we start by noting that value of a formula φ according to some valuation v is determined only by the values associated to a finite set of propositional variables $\mathbf{X}$ such that $\text{Var}(\varphi) \subset \mathbf{X}$; this very property is the crux for the possibility that logical formulas represent functions. We next generalize this notion.

Definition 3.1 Let φ be a formula and let Φ be a set of formulas. We say that a set of variables $\mathbf{X}_n$ *determines* φ *modulo* Φ*-satisfiable* if:

- For any $\langle x_1, \ldots, x_n \rangle \in [0,1]^n$, there exists at least one valuation $v \in \mathbf{Val}_\Phi$, such that $v(X_j) = x_j$, for $j = 1, \ldots, n$;
- For any valuations $v, v' \in \mathbf{Val}_\Phi$, such that $v(X_j) = v'(X_j)$, for $j = 1, \ldots, n$, $v(\varphi) = v'(\varphi)$.

For instance, for any formula φ such that $\mathrm{Var}(\varphi) \subset \mathbf{X}_n$, $\mathbf{X}_n$ determines φ modulo $\varnothing$-satisfiable, by truth functionality and the fact that $\mathbf{Val}_\varnothing = \mathbf{Val}$.

It is important to note that any set $\mathbf{X}_n$ can now represent a rational fraction $\frac{1}{d}$ by determining a propositional variable $Z_{\frac{1}{d}}$ modulo $\varphi_{\frac{1}{d}} = Z_{\frac{1}{d}} \leftrightarrow \neg(d-1)Z_{\frac{1}{d}}$ satisfiable, with $d \in \mathbb{N}^*$. In fact, for any valuation $v \in \mathbf{Val}$, if $v(\varphi_{\frac{1}{d}}) = 1$, then $v(Z_{\frac{1}{d}}) = \frac{1}{d}$. We define representation modulo satisfiability in a way that retrieves property (3).

Definition 3.2 Let $f : [0,1]^n \to [0,1]$ be a function, and $\langle \varphi, \Phi \rangle$ be a pair where φ is a formula and Φ is a set of formulas. We say that φ *represents* f *modulo* Φ*-satisfiable* or that $\langle \varphi, \Phi \rangle$ *represents* f *in the system* $Ł_\infty$*-MODSAT* if:

- $\mathbf{X}_n$ determines φ modulo Φ-satisfiable;
- $f(v(X_1), \ldots, v(X_n)) = v(\varphi)$, for $v \in \mathbf{Val}_\Phi$.

Representation modulo satisfiability presented in [10] has a different approach, which we call function-based and is more restrictive than the one presented here, which we call formula-based. However, the representation methods and algorithms we develop in this work apply to both approaches.

4 Rational McNaughton Functions

Our algorithm uses a lattice representation of rational McNaughton functions; before that we employ an encoding based in [17,18] as follows. Let Ω° be the interior of a set $\Omega \subset \mathbb{R}^n$. A rational McNaughton function $f : [0,1]^n \to [0,1]$ is given by m (not necessarily distinct) linear pieces

$$p_i(\mathbf{x}) = \gamma_{i0} + \gamma_{i1}x_1 + \cdots + \gamma_{in}x_n, \tag{4}$$

for $\mathbf{x} = \langle x_1, \ldots, x_n \rangle \in [0,1]^n$, $\gamma_{ij} \in \mathbb{Q}$ and $i = 1, \ldots, m$, with each linear piece p_i identical to f over a convex set $\Omega_i \subset [0,1]^n$ called *region* such that:

- $\bigcup_{i=1}^m \Omega_i = [0,1]^n$;
- $\Omega_{i'}^\circ \cap \Omega_{i''}^\circ = \varnothing$, for $i' \neq i''$;
- Regions Ω_i are given in such a way that there is a polynomial procedure to determine whether or not a linear piece p_k is *above* other linear piece p_i over region Ω_i, that is whether or not $p_k(\mathbf{x}) \geq p_i(\mathbf{x})$, for all $\mathbf{x} \in \Omega_i$.

A rational McNaughton function as above is said to be in *regional format*. Note that

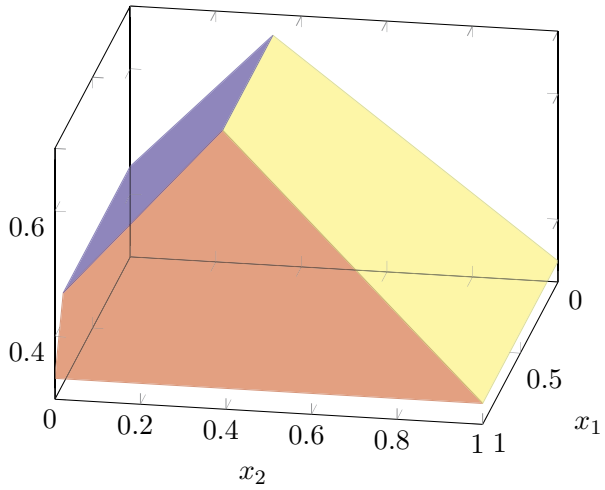

Fig. 2. Graph of rational McNaughton function with three linear pieces over $[0,1]^2$.

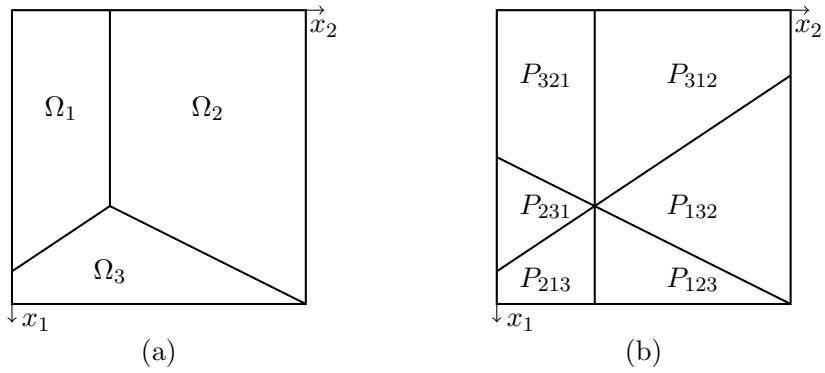

Fig. 3. Some possible region configurations for function f in Example 4.1.

in this format the number of linear pieces is equal to the number of regions; in this case, the size of a function is the sum of the number of bits necessary to represent its linear pieces coefficients as fractions $\frac{a}{b}$ plus the space necessary for representing its regions in some assumed encoding.

Example 4.1 Rational McNaughton function f with graph in Figure 2 may be given by the linear pieces:

- $p_1(x_1, x_2) = \frac{4}{9} + \frac{2}{3}x_2$;
- $p_2(x_1, x_2) = \frac{5}{6} - \frac{1}{2}x_2$;
- $p_3(x_1, x_2) = \frac{4}{3} - x_1$.

Regions associated to each linear piece are depicted in Figure 3(a). Determining if some linear piece p_k is above other linear piece p_i over Ω_i is equivalent to determining if the system of corresponding inequalities in Table 1 with added equation $p_k - p_i = 0$ has no solution and $p_k(\mathbf{x}_0) > p_i(\mathbf{x}_0)$, for some point $\mathbf{x}_0 \in \Omega_i^\circ$, a tractable process.

Let $f : [0,1]^n \to [0,1]$ be a rational McNaughton function as defined in Section 2,

Ω_1°	Ω_2°	Ω_3°
$8 - 9x_1 - 6x_2 > 0$	$1 - 2x_1 + x_2 > 0$	$-8 + 9x_1 + 6x_2 > 0$
$\frac{1}{3} - x_2 > 0$	$-\frac{1}{3} + x_2 > 0$	$-1 + 2x_1 - x_2 > 0$
$x_1 > 0$	$x_1 > 0$	$1 - x_1 > 0$
$x_2 > 0$	$1 - x_2 > 0$	$x_2 > 0$

Table 1
Sets Ω_i° for function f in Example 4.1.

with distinct linear pieces $p_1, \ldots, p_m$. For each permutation ρ of the set $\{1, \ldots, m\}$, we define the polyhedron

$$P_\rho = \{\mathbf{x} \in [0,1]^n \mid p_{\rho(1)}(\mathbf{x}) \geq \cdots \geq p_{\rho(m)}(\mathbf{x})\}. \tag{5}$$

Let $\mathcal{C}$ be the set of n-dimensional polyhedra P_ρ, for some permutation ρ.

Proposition 4.2 *The set $\mathcal{C}$ has the following properties:*

(a) $\bigcup \mathcal{C} = [0,1]^n$;

(b) For polyhedron $P \in \mathcal{C}$ and indexes $i', i'' \in \{1, \ldots, m\}$ with $i' \neq i''$, $p_{i'}(\mathbf{x}) \neq p_{i''}(\mathbf{x})$, for any $\mathbf{x} \in P^\circ$;

(c) $P'^\circ \cap P''^\circ = \varnothing$, for $P', P'' \in \mathcal{C}$ such that $P' \neq P''$;

(d) For each polyhedron $P \in \mathcal{C}$, there is an index $i_P \in \{1, \ldots, m\}$ such that $f(\mathbf{x}) = p_{i_P}(\mathbf{x})$, for $\mathbf{x} \in [0,1]^n$.

Proof.

(a) For any $\mathbf{x} \in P \in \mathcal{C}$, $\mathbf{x} \in [0,1]^n$. On the other hand, for any $\mathbf{x} \in [0,1]^n$, there is a permutation ρ for which P_ρ is n-dimensional and $\mathbf{x} \in P_\rho$.

(b) Let $\mathbf{x} \in P^\circ$ and let $i', i'' \in \{1, \ldots, m\}$ be indexes such that $i' \neq i''$. Since $p_{i'}$ and $p_{i''}$ are linear pieces, if $p_{i'}(\mathbf{x}) = p_{i''}(\mathbf{x})$, for some $\mathbf{x} \in P^\circ$, there would be points $\mathbf{x}_1, \mathbf{x}_2 \in P^\circ$ in a neighborhood of $\mathbf{x}$ such that $p_{i'}(\mathbf{x}_1) < p_{i''}(\mathbf{x}_1)$ and $p_{i''}(\mathbf{x}_2) < p_{i'}(\mathbf{x}_2)$, contrary to the definition of P.

(c) Let $\mathbf{x} \in P'^\circ \cap P''^\circ$. Then, by definitions of P' and P'', there are $i', i'' \in \{1, \ldots, m\}$ such that $p_{i'}(\mathbf{x}) = p_{i''}(\mathbf{x})$, contrary to item (b).

(d) Let $\{i_1, \ldots, i_k\} \subset \{1, \ldots, m\}$ be a non-singleton set of indexes such that, for any $\mathbf{x} \in P^\circ$, there is $l \in \{1, \ldots, k\}$, such that $f(\mathbf{x}) = p_{i_l}(\mathbf{x})$ and $U_{i_l} = \{\mathbf{x} \in P^\circ \mid f(\mathbf{x}) = p_{i_l}(\mathbf{x})\} \neq \varnothing$, for $l = 1, \ldots, k$. We have that $\cup_{l=1}^k U_{i_l} = P^\circ$ and, by item (b), $U_{i_{l'}} \cap U_{i_{l''}} = \varnothing$, for $l' \neq l''$. As P° is a connected set, there are distinct $i', i'' \in \{i_1, \ldots, i_k\}$ and $\mathbf{b} \in P^\circ$ such that $\mathbf{b} \in \partial U_{i'}$ and $\mathbf{b} \in U_{i''}$. As $p_{i'}$ restricted to $U_{i'} \cup \{\mathbf{b}\}$ is continuous, for any sequence $\{\mathbf{b}_n\} \subset U_{i'}$ such that $\lim \mathbf{b}_n = \mathbf{b}$ (which exists since $\mathbf{b} \in \partial U_{i'}$), we have that $\lim f(\mathbf{b}_n) = \lim p_{i'}(\mathbf{b}_n) = p_{i'}(\mathbf{b})$. However, $f(\mathbf{b}) = p_{i''}(\mathbf{b}) \neq p_{i'}(\mathbf{b})$, by item (b), contrary to the continuity of f.

□

Polyhedra in $\mathcal{C}$ may play the role of regions since they are convex sets with the properties above; determining whether a linear piece p_k is above other linear piece p_i over $P \in \mathcal{C}$ comes down to comparing their values for some point $\mathbf{x} \in P^\circ$. Note that the same linear piece p_i may be associated to many distinct polyhedra. Thus, any rational McNaughton function may be encoded in regional format. Figure 3(b) shows the polyhedra-based configuration $\mathcal{C}$ for the function in Example 4.1.

The setback with describing a rational McNaughton function using the set $\mathcal{C}$ of polyhedra is that in the worst case $|\mathcal{C}| = m!$. However, in general there are smaller sets of regions that comply with representation restrictions above [18].

5 A Particular Case: Truncated Linear Functions

Let us show the possibility of representing a rational McNaughton function modulo satisfiability and develop a polynomial algorithm for computing such representation in the particular case that function is a truncated linear polynomial with rational coefficients.

Let $p : [0,1]^n \to \mathbb{R}$ be a nonzero linear polynomial given by

$$p(\mathbf{x}) = \frac{a_0}{b_0} + \frac{a_1}{b_1}x_1 + \cdots + \frac{a_n}{b_n}x_n, \tag{6}$$

for $\mathbf{x} = \langle x_1, \ldots, x_n \rangle \in [0,1]^n$, $a_j \in \mathbb{Z}$, and $b_j \in \mathbb{Z}_+^*$. We want to build a representation for the function $p^\# : [0,1]^n \to [0,1]$ given by

$$p^\#(\mathbf{x}) = \min\Big(1, \max\big(0, p(\mathbf{x})\big)\Big). \tag{7}$$

We have that $p^\#(\mathbf{x}) = 0$, if $p(\mathbf{x}) < 0$; $p^\#(\mathbf{x}) = 1$, if $p(\mathbf{x}) > 1$; and $p^\#(\mathbf{x}) = p(\mathbf{x})$, otherwise.

In order to rewrite expression (6), we define:

$$\begin{aligned} &\alpha_j = a_j, \text{ for } j \in P; \\ &\alpha_j = -a_j, \text{ for } j \in N; \\ &\beta_j = \beta \cdot b_j, \text{ for } j = 0, \ldots, n; \end{aligned}$$

where $j \in P$, if $a_j > 0$, and $j \in N$, if $a_j < 0$, with $P \cup N \subset \{0, \ldots, n\}$, and β is the least integer greater than or equal to

$$\max\left\{\sum_{j\in P}\frac{a_j}{b_j}, \quad -\sum_{j\in N}\frac{a_j}{b_j}\right\}.$$

We have that $\alpha_j \in \mathbb{Z}_+$ and $\beta_j \in \mathbb{Z}_+^*$, for $j = 0, \ldots, n$. Let $x_0 = 1$ and define functions $p_P : [0,1]^n \to \mathbb{R}$ and $p_N : [0,1]^n \to \mathbb{R}$, for $\mathbf{x} = \langle x_1, \ldots, x_n \rangle \in [0,1]^n$, by:

$$p_P(\mathbf{x}) = \sum_{j\in P}\frac{\alpha_j}{\beta_j}x_j; \qquad p_N(\mathbf{x}) = \sum_{j\in N}\frac{\alpha_j}{\beta_j}x_j. \tag{8}$$

Lemma 5.1 *Functions p, p_P, and p_N in (6) and (8) have the following properties, for $\mathbf{x} \in [0,1]^n$:*

(a) $p(\mathbf{x}) = \beta \cdot \big(p_P(\mathbf{x}) - p_N(\mathbf{x})\big)$;

(b) $0 \leq p_P(\mathbf{x}), p_N(\mathbf{x}) \leq 1$.

Proof. By elementary algebraic manipulation. □

Lemma above decomposes function p in terms of p_P and p_N, let us represent the latter ones. Let $Z^p_j, Z_{\frac{1}{\beta_j}} \in \mathbb{P}$. For a set of indexes $J \in \{P, N\}$, define:

$$\tilde{\varphi}_J = \bigoplus_{j \in J \setminus \{0\}} \alpha_j Z^p_j; \qquad \tilde{\Phi}_J = \bigcup_{j \in J \setminus \{0\}} \left\{ \varphi_{\frac{1}{\beta_j}}, \quad \beta_j Z^p_j \leftrightarrow X_j, \quad Z^p_j \to Z_{\frac{1}{\beta_j}} \right\}.$$

And then, define:

$$\begin{array}{lll} \bar{\varphi}_J = \tilde{\varphi}_J; & \bar{\Phi}_J = \tilde{\Phi}_J, & \text{if } 0 \notin J; \\ \bar{\varphi}_J = \alpha_0 Z_{\frac{1}{\beta_0}} \oplus \tilde{\varphi}_J; & \bar{\Phi}_J = \tilde{\Phi}_J \cup \{\varphi_{\frac{1}{\beta_0}}\}, & \text{otherwise.} \end{array} \tag{9}$$

Lemma 5.2 *Functions p_P and p_N in (8) may respectively be represented by $\langle \bar{\varphi}_P, \bar{\Phi}_P \rangle$ and $\langle \bar{\varphi}_N, \bar{\Phi}_N \rangle$ in (9).*

Proof. Let $J \in \{P, N\}$. If $J = \varnothing$, then $\langle \bar{\varphi}_J, \bar{\Phi}_J \rangle = \langle \mathbf{0}, \varnothing \rangle$ represents p_J. For $\langle x_1, \ldots, x_n \rangle \in [0,1]^n$, define a valuation $v \in \mathbf{Val}$ such that $v(X_j) = x_j$ and $v(Z^p_j) = \frac{x_j}{\beta_j}$, for $j \in J \setminus \{0\}$, and $v(Z_{\frac{1}{\beta_j}}) = \frac{1}{\beta_j}$, for $j \in J$. We have that $v \in \mathbf{Val}_{\bar{\Phi}_J}$. Now, let $v, v' \in \mathbf{Val}_{\bar{\Phi}_J}$ such that $v(X_j) = v'(X_j)$, for $j = 1, \ldots, n$. By rational constant representation, $v(Z_{\frac{1}{\beta_j}}) = v'(Z_{\frac{1}{\beta_j}}) = \frac{1}{\beta_j}$, for $j \in J$. Thus $v(Z^p_j) \leq \frac{1}{\beta_j}$ and $v'(Z^p_j) \leq \frac{1}{\beta_j}$, which implies that $\beta_j \cdot v(Z^p_j) = v(\beta_j Z^p_j) = v(X_j) = v'(X_j) = v'(\beta_j Z^p_j) = \beta_j \cdot v'(Z^p_j)$ and, then, $v(Z^p_j) = v'(Z^p_j)$, for $j \in J \setminus \{0\}$. Therefore, $v(\bar{\varphi}_J) = v'(\bar{\varphi}_J)$ and $\mathbf{X}_n$ determines $\bar{\varphi}_J$ modulo $\bar{\Phi}_J$-satisfiable. Finally, suppose $v \in \mathbf{Val}_{\bar{\Phi}_J}$. In the case where $0 \in J$,

$$p_J(v(X_1), \ldots, v(X_n)) = \alpha_0 \cdot v(Z_{\frac{1}{\beta_0}}) + \sum_{j \in J \setminus \{0\}} \alpha_j \cdot v(Z^p_j) = v(\bar{\varphi}_J),$$

by Lemma 5.1 and aforementioned equations $v(Z_{\frac{1}{\beta_0}}) = \frac{1}{\beta_0}$ and $\beta_j \cdot v(Z^p_j) = v(X_j)$. The case where $0 \notin J$ is similar. □

For the final step towards a representation for $p^{\#}$, we define:

$$\bar{\varphi}_p = \beta[\neg(\bar{\varphi}_P \to \bar{\varphi}_N)]; \qquad \bar{\Phi}_p = \bar{\Phi}_P \cup \bar{\Phi}_N. \tag{10}$$

Theorem 5.3 *Function $p^{\#}$ in (7) may be represented by $\langle \bar{\varphi}_p, \bar{\Phi}_p \rangle$ in (10).*

Proof. For $\langle x_1, \ldots, x_n \rangle \in [0,1]^n$, there exists $v \in \mathbf{Val}_{\bar{\Phi}_p}$ such that $v(X_j) = x_j$ as in the proof of Lemma 5.2 with $J = P \cup N$. Now, let $v, v' \in \mathbf{Val}_{\bar{\Phi}_p}$ such that

$v(X_j) = v'(X_j)$, for $j = 1, \ldots, n$. In particular, $v, v' \in \mathbf{Val}_{\bar{\Phi}_J}$ and, by Lemma 5.2, $v(\bar{\varphi}_J) = v'(\bar{\varphi}_J)$, for $J \in \{P, N\}$. Therefore, $v(\bar{\varphi}_p) = v'(\bar{\varphi}_p)$ and $\mathbf{X}_n$ determines $\bar{\varphi}_p$ modulo $\bar{\Phi}_p$-satisfiable. Finally, suppose $v \in \mathbf{Val}_{\bar{\Phi}_p}$. In particular, $v \in \mathbf{Val}_{\bar{\Phi}_P}$ and $v \in \mathbf{Val}_{\bar{\Phi}_N}$. If $p(v(X_1), \ldots, v(X_n)) \leq 0$, by Lemma 5.1, $p_P(v(X_1), \ldots, v(X_n)) \leq p_N(v(X_1), \ldots, v(X_n))$. Therefore, by Lemma 5.2, $v(\bar{\varphi}_P) \leq v(\bar{\varphi}_N)$ and, then, $v(\bar{\varphi}_p) = 0$. On the other hand, if $p(v(X_1), \ldots, v(X_n)) \geq 0$, by Lemma 5.1, $p_P(v(X_1), \ldots, v(X_n)) \geq p_N(v(X_1), \ldots, v(X_n))$. Therefore, by Lemma 5.2, $v(\bar{\varphi}_P) \geq v(\bar{\varphi}_N)$ and, then, $v(\neg(\bar{\varphi}_P \to \bar{\varphi}_N)) = 1 - \min(1, 1 - v(\bar{\varphi}_P) + v(\bar{\varphi}_N)) = v(\bar{\varphi}_P) - v(\bar{\varphi}_N)$. Finally, by Lemmas 5.1 and 5.2, $p(v(X_1), \ldots, v(X_n)) = \beta \cdot (v(\bar{\varphi}_P) - v(\bar{\varphi}_N))$, hence $p^{\#}(v(X_1), \ldots, v(X_n)) = v(\bar{\varphi}_p)$ in any case. □

Table 2 shows how functions in Example 4.1 can be represented as in Theorem 5.3.

In order to set up a polynomial algorithm for computing a representation $\langle \varphi_p, \Phi_p \rangle$ for $p^{\#}$, we analyze more closely expressions $n\psi$, which show up in $\bar{\varphi}_p$ and in formulas in $\bar{\Phi}_p$. These expressions are exponential in the binary representation of n since it denotes an n-fold repetition of formula ψ. We deviate from this situation by using $\lfloor \log n \rfloor + 1$ new propositional variables $\xi^0_\psi, \xi^1_\psi, \ldots, \xi^{\lfloor \log n \rfloor}_\psi$ and replacing every occurrence of $n\psi$, where $n \in \mathbb{N} \setminus \{0, 1\}$, with the formula

$$\xi_{n\psi} =_{\text{def}} \bigoplus_{\substack{k=0 \\ n_k=1}}^{\lfloor \log n \rfloor} \xi^k_\psi, \tag{11}$$

where $n_k \in \{0, 1\}$ comes from the binary representation $\sum_{k=0}^{\lfloor \log n \rfloor} 2^k n_k$ of n, and by adding the following formulas to $\bar{\Phi}_p$:

$$\begin{aligned} &\xi^0_\psi \leftrightarrow \psi; \\ &\xi^k_\psi \leftrightarrow \xi^{k-1}_\psi \oplus \xi^{k-1}_\psi, \text{ for } k = 1, \ldots, \lfloor \log n \rfloor. \end{aligned} \tag{12}$$

These formulas define the propositional variables ξ^k_ψ and we call $\Xi_{n\psi}$ the set that comprehends them. In this way we avoid exponential blow up as shown in Theorem 5.5.

Lemma 5.4 *Let $n \in \mathbb{N} \setminus \{0, 1\}$, ψ be a formula, and $\xi_{n\psi}$ and $\Xi_{n\psi}$ be respectively a formula as in (11) and a set as in (12) built from n and ψ. For any valuation $v \in \mathbf{Val}_{\Xi_{n\psi}}$, $v(n\psi) = v(\xi_{n\psi})$.*

$\bar{\varphi}_{p_1}$:	$\neg\left(Z_{\frac{1}{18}} \oplus Z_{\frac{1}{18}} \oplus Z_{\frac{1}{18}} \oplus Z_{\frac{1}{18}} \oplus Z_2^{p_1} \oplus Z_2^{p_1} \to \mathbf{0}\right)$ $\oplus\neg\left(Z_{\frac{1}{18}} \oplus Z_{\frac{1}{18}} \oplus Z_{\frac{1}{18}} \oplus Z_{\frac{1}{18}} \oplus Z_2^{p_1} \oplus Z_2^{p_1} \to \mathbf{0}\right)$
$\bar{\Phi}_{p_1}$:	$Z_{\frac{1}{18}} \leftrightarrow \neg\Big(Z_{\frac{1}{18}} \oplus Z_{\frac{1}{18}} \oplus Z_{\frac{1}{18}} \oplus Z_{\frac{1}{18}} \oplus Z_{\frac{1}{18}} \oplus Z_{\frac{1}{18}} \oplus Z_{\frac{1}{18}} \oplus Z_{\frac{1}{18}} \oplus Z_{\frac{1}{18}}$ $\oplus Z_{\frac{1}{18}} \oplus Z_{\frac{1}{18}} \oplus Z_{\frac{1}{18}} \oplus Z_{\frac{1}{18}} \oplus Z_{\frac{1}{18}} \oplus Z_{\frac{1}{18}} \oplus Z_{\frac{1}{18}} \oplus Z_{\frac{1}{18}}\Big)$
	$Z_{\frac{1}{6}} \leftrightarrow \neg\left(Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}}\right)$
	$Z_2^{p_1} \oplus Z_2^{p_1} \oplus Z_2^{p_1} \oplus Z_2^{p_1} \oplus Z_2^{p_1} \oplus Z_2^{p_1} \leftrightarrow X_2$
	$Z_2^{p_1} \to Z_{\frac{1}{6}}$
$\bar{\varphi}_{p_2}$:	$\neg\left(Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}} \to Z_2^{p_2}\right)$
$\bar{\Phi}_{p_2}$:	$Z_{\frac{1}{6}} \leftrightarrow \neg\left(Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}}\right)$
	$Z_{\frac{1}{2}} \leftrightarrow \neg Z_{\frac{1}{2}}$
	$Z_2^{p_2} \oplus Z_2^{p_2} \leftrightarrow X_2$
	$Z_2^{p_2} \to Z_{\frac{1}{2}}$
$\bar{\varphi}_{p_3}$:	$\neg\left(Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}} \to Z_1^{p_3}\right) \oplus \neg\left(Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}} \to Z_1^{p_3}\right)$
$\bar{\Phi}_{p_3}$:	$Z_{\frac{1}{6}} \leftrightarrow \neg\left(Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}} \oplus Z_{\frac{1}{6}}\right)$
	$Z_{\frac{1}{2}} \leftrightarrow \neg Z_{\frac{1}{2}}$
	$Z_1^{p_3} \oplus Z_1^{p_3} \leftrightarrow X_1$
	$Z_1^{p_3} \to Z_{\frac{1}{2}}$

Table 2
Representations as in (10) for functions $p_1^{\#}$, $p_2^{\#}$ and $p_3^{\#}$, where functions p_1, p_2 and p_3 are from Example 4.1.

Proof. For $v \in \mathbf{Val}_{\Xi_{n\psi}}$ and $k = 0, \ldots, \lfloor \log n \rfloor$, $v(\xi^k_\psi) = \min(1, 2^k v(\psi))$. Then,

$$v(n\psi) = \min\left(1, \sum_{k=0}^{\lfloor \log n \rfloor} 2^k n_k v(\psi)\right)$$
$$= \min\left(1, \sum_{k=0}^{\lfloor \log n \rfloor} v(\xi^k_\psi) n_k\right) = v\left(\bigoplus_{n_k=1} \xi^k_\psi\right) = v(\xi_{n\psi}),$$

where $n_k \in \{0,1\}$ in the binary representation $n = \sum_{k=0}^{\lfloor \log n \rfloor} 2^k n_k$. □

Theorem 5.5 *Let $n \in \mathbb{N} \setminus \{0,1\}$, ψ be a formula, and $\langle \varphi_p, \Phi_p \rangle$ be a pair defined from representation $\langle \bar{\varphi}_p, \bar{\Phi}_p \rangle$ in (10) by replacing any occurrence of $n\psi$ in $\bar{\varphi}_p$ and $\bar{\Phi}_p$ with $\xi_{n\psi}$ in (11) and by adding formulas in set $\Xi_{n\psi}$ in (12) to $\bar{\Phi}_p$. Then, $\langle \varphi_p, \Phi_p \rangle$ is also a representation for $p^\#$ in (7). Furthermore, $\langle \varphi_p, \Phi_p \rangle$ is a representation for $p^\#$ even if it is defined by multiple suitable replacements of expressions $n_l\psi_l$, for $l = 1, \ldots, L$.*

Proof. For $\langle x_1, \ldots, x_n \rangle \in [0,1]^n$, define a valuation v such that $v(X_j) = x_j$ and $v(Z^p_j) = \frac{x_j}{\beta_j}$, for $j = 1, \ldots, n$, $v(Z_{\frac{1}{\beta_j}}) = \frac{1}{\beta_j}$, for $j = 0, \ldots, n$, $v(\xi^0_\psi) = v(\psi)$, and $v(\xi^k_\psi) = \min(1, v(\xi^{k-1}_\psi) + v(\xi^{k-1}_\psi))$, for $k = 1, \ldots, \lfloor \log n \rfloor$. Note that $v \in \mathbf{Val}_{\bar{\Phi}_p}$ and $v \in \mathbf{Val}_{\Xi_{n\psi}}$, then, by Lemma 5.4, as $\Xi_{n\psi} \subset \Phi_p$, we have that $v \in \mathbf{Val}_{\Phi_p}$. Still, for any $v \in \Phi_p$, we have that $v \in \mathbf{Val}_{\Xi_{n\psi}}$ and, by Lemma 5.4, $v \in \bar{\Phi}_p$. Therefore, again by Lemma 5.4, for $v, v' \in \Phi_p$ such that $v(X_j) = v'(X_j)$, for $j = 1, \ldots, n$, it follows that $v(\varphi_p) = v'(\varphi_p)$, $\mathbf{X}_n$ determines φ_p modulo Φ_p-satisfiable, and $p^\#(v(X_1), \ldots, v(X_n)) = v(\varphi_p)$. This argument still holds when considering multiple replacements. □

We set $\langle \varphi_p, \Phi_p \rangle$ from $\langle \bar{\varphi}_p, \bar{\Phi}_p \rangle$ in (10) by properly replacing all occurrences of $n_l\psi_l$ as stated in the above theorem. By construction, $\langle \varphi_p, \Phi_p \rangle$ is given by

$$\varphi_p = \beta[\neg(\varphi_P \to \varphi_N)]; \qquad \Phi_p = \Phi_P \cup \Phi_N; \tag{13}$$

where φ_P, φ_N, Φ_P, and Φ_N are properly defined from their barred correspondents in (9). Table 3 shows how functions in Example 4.1 can be represented as in Theorem 5.5.

Algorithms 1 and 2 compute the representation modulo satisfiability of $n\psi$. Algorithm 1 returns $\mathbf{0}$ and ψ in the limit cases $n = 0$ and $n = 1$ (lines 1 to 5); when $n \in \mathbb{N} \setminus \{0,1\}$, it returns formula $\xi_{n\psi}$ in (11) by building it in line 6 plus a $\lfloor \log n \rfloor + 1$ iteration loop (lines 7 to 13) where the n_k's in the binary representation of n are calculated by the routine in lines 8 and 9. Algorithm 2 returns $\varnothing$ in the limit cases $n = 0$ and $n = 1$ (lines 1 to 3); when $n \in \mathbb{N} \setminus \{0,1\}$, it returns set $\Xi_{n\psi}$ that comprehends formulas (12) by building it in line 4 plus a $\lfloor \log n \rfloor$ iteration loop (lines 5 to 7). Both algorithms terminate in time $O(\log n)$ assuming propositional variables are all represented with a constant size.

$\varphi_{p_1}: \xi^1_{\neg(\xi^2_{Z_{\frac{1}{18}}} \oplus \xi^1_{Z_2^{p_1}} \to \mathbf{0})}$	
$\Phi_{p_1}: Z_{\frac{1}{18}} \leftrightarrow \neg\left(\xi^4_{Z_{\frac{1}{18}}} \oplus \xi^0_{Z_{\frac{1}{18}}}\right)$	$\xi^0_{Z_2^{p_1}} \leftrightarrow Z_2^{p_1}$
$\xi^0_{Z_{\frac{1}{18}}} \leftrightarrow Z_{\frac{1}{18}}$	$\xi^1_{Z_2^{p_1}} \leftrightarrow \xi^0_{Z_2^{p_1}} \oplus \xi^0_{Z_2^{p_1}}$
$\xi^1_{Z_{\frac{1}{18}}} \leftrightarrow \xi^0_{Z_{\frac{1}{18}}} \oplus \xi^0_{Z_{\frac{1}{18}}}$	$\xi^2_{Z_2^{p_1}} \leftrightarrow \xi^1_{Z_2^{p_1}} \oplus \xi^1_{Z_2^{p_1}}$
$\xi^2_{Z_{\frac{1}{18}}} \leftrightarrow \xi^1_{Z_{\frac{1}{18}}} \oplus \xi^1_{Z_{\frac{1}{18}}}$	$\xi^0_{Z_{\frac{1}{6}}} \leftrightarrow Z_{\frac{1}{6}}$
$\xi^3_{Z_{\frac{1}{18}}} \leftrightarrow \xi^2_{Z_{\frac{1}{18}}} \oplus \xi^2_{Z_{\frac{1}{18}}}$	$\xi^1_{Z_{\frac{1}{6}}} \leftrightarrow \xi^0_{Z_{\frac{1}{6}}} \oplus \xi^0_{Z_{\frac{1}{6}}}$
$\xi^4_{Z_{\frac{1}{18}}} \leftrightarrow \xi^3_{Z_{\frac{1}{18}}} \oplus \xi^3_{Z_{\frac{1}{18}}}$	$\xi^2_{Z_{\frac{1}{6}}} \leftrightarrow \xi^1_{Z_{\frac{1}{6}}} \oplus \xi^1_{Z_{\frac{1}{6}}}$
$Z_{\frac{1}{6}} \leftrightarrow \neg\left(\xi^2_{Z_{\frac{1}{6}}} \oplus \xi^0_{Z_{\frac{1}{6}}}\right)$	$\xi^0_{\neg(\xi^2_{Z_{\frac{1}{18}}} \oplus \xi^1_{Z_2^{p_1}} \to \mathbf{0})} \leftrightarrow \neg\left(\xi^2_{Z_{\frac{1}{18}}} \oplus \xi^1_{Z_2^{p_1}} \to \mathbf{0}\right)$
$\xi^2_{Z_2^{p_1}} \oplus \xi^1_{Z_2^{p_1}} \leftrightarrow X_2$	$\xi^1_{\neg(\xi^2_{Z_{\frac{1}{18}}} \oplus \xi^1_{Z_2^{p_1}} \to \mathbf{0})} \leftrightarrow \xi^0_{\neg(\xi^2_{Z_{\frac{1}{18}}} \oplus \xi^1_{Z_2^{p_1}} \to \mathbf{0})} \oplus \xi^0_{\neg(\xi^2_{Z_{\frac{1}{18}}} \oplus \xi^1_{Z_2^{p_1}} \to \mathbf{0})}$
$Z_2^{p_1} \to Z_{\frac{1}{6}}$	
$\varphi_{p_2}: \neg\left(\xi^2_{Z_{\frac{1}{6}}} \oplus \xi^0_{Z_{\frac{1}{6}}} \to Z_2^{p_2}\right)$	
$\Phi_{p_2}: Z_{\frac{1}{6}} \leftrightarrow \neg\left(\xi^2_{Z_{\frac{1}{6}}} \oplus \xi^0_{Z_{\frac{1}{6}}}\right)$	$\xi^1_{Z_{\frac{1}{6}}} \leftrightarrow \xi^0_{Z_{\frac{1}{6}}} \oplus \xi^0_{Z_{\frac{1}{6}}}$
$Z_{\frac{1}{2}} \leftrightarrow \neg Z_{\frac{1}{2}}$	$\xi^2_{Z_{\frac{1}{6}}} \leftrightarrow \xi^1_{Z_{\frac{1}{6}}} \oplus \xi^1_{Z_{\frac{1}{6}}}$
$\xi^1_{Z_2^{p_2}} \leftrightarrow X_2$	$\xi^0_{Z_2^{p_2}} \leftrightarrow Z_2^{p_2}$
$Z_2^{p_2} \to Z_{\frac{1}{2}}$	$\xi^1_{Z_2^{p_2}} \leftrightarrow \xi^0_{Z_2^{p_2}} \oplus \xi^0_{Z_2^{p_2}}$
$\xi^0_{Z_{\frac{1}{6}}} \leftrightarrow Z_{\frac{1}{6}}$	
$\varphi_{p_3}: \xi^1_{\neg(\xi^2_{Z_{\frac{1}{6}}} \to Z_1^{p_3})}$	
$\Phi_{p_3}: Z_{\frac{1}{6}} \leftrightarrow \neg\left(\xi^2_{Z_{\frac{1}{6}}} \oplus \xi^0_{Z_{\frac{1}{6}}}\right)$	$Z_1^{p_3} \to Z_{\frac{1}{2}}$
$\xi^0_{Z_{\frac{1}{6}}} \leftrightarrow Z_{\frac{1}{6}}$	$\xi^0_{Z_1^{p_3}} \leftrightarrow Z_1^{p_3}$
$\xi^1_{Z_{\frac{1}{6}}} \leftrightarrow \xi^0_{Z_{\frac{1}{6}}} \oplus \xi^0_{Z_{\frac{1}{6}}}$	$\xi^1_{Z_1^{p_3}} \leftrightarrow \xi^0_{Z_1^{p_3}} \oplus \xi^0_{Z_1^{p_3}}$
$\xi^2_{Z_{\frac{1}{6}}} \leftrightarrow \xi^1_{Z_{\frac{1}{6}}} \oplus \xi^1_{Z_{\frac{1}{6}}}$	$\xi^0_{\neg(\xi^2_{Z_{\frac{1}{6}}} \to Z_1^{p_3})} \leftrightarrow \neg\left(\xi^2_{Z_{\frac{1}{6}}} \to Z_1^{p_3}\right)$
$Z_{\frac{1}{2}} \leftrightarrow \neg Z_{\frac{1}{2}}$	$\xi^1_{\neg(\xi^2_{Z_{\frac{1}{6}}} \to Z_1^{p_3})} \leftrightarrow \xi^0_{\neg(\xi^2_{Z_{\frac{1}{6}}} \to Z_1^{p_3})} \oplus \xi^0_{\neg(\xi^2_{Z_{\frac{1}{6}}} \to Z_1^{p_3})}$
$\xi^1_{Z_1^{p_3}} \leftrightarrow X_1$	

Table 3
Representations as in (13) for functions $p_1^{\#}$, $p_2^{\#}$ and $p_3^{\#}$, where functions p_1, p_2 and p_3 are from Example 4.1.

Algorithm 1 BINARY-F: computes formula $\xi_{n\psi}$ in (11) or $\mathbf{0}$ or ψ

Input: A natural number n and a formula ψ.
Output: Formula $\xi_{n\psi}$.

1: **if** $n = 0$ **then**
2: **return** $\mathbf{0}$;
3: **else if** $n = 1$ **then**
4: **return** ψ;
5: **end if**
6: $q := n$, $n_k := 0$, $\xi_{n\psi} := \mathbf{0}$;
7: **for** $k = 0, \ldots, \lfloor \log n \rfloor$ **do**
8: $n_k :=$ remainder from division of q by 2;
9: $q :=$ quotient from division of q by 2;
10: **if** $n_k = 1$ **then**
11: $\xi_{n\psi} := \xi_\psi^k \oplus \xi_{n\psi}$;
12: **end if**
13: **end for**
14: **return** $\xi_{n\psi}$;

Algorithm 2 BINARY-S: computes set $\Xi_{n\psi}$ in (12) or $\varnothing$

Input: A natural number n and a formula ψ.
Output: Set $\Xi_{n\psi}$.

1: **if** $n = 0$ or $n = 1$ **then**
2: **return** $\varnothing$;
3: **end if**
4: $\Xi_{n\psi} := \{\xi_\psi^0 \leftrightarrow \psi\}$;
5: **for** $k = 1, \ldots, \lfloor \log n \rfloor$ **do**
6: $\Xi_{n\psi} := \Xi_{n\psi} \cup \{\xi_\psi^k \leftrightarrow \xi_\psi^{k-1} \oplus \xi_\psi^{k-1}\}$;
7: **end for**
8: **return** $\Xi_{n\psi}$;

Algorithm 3 computes a representation modulo satisfiability for $p^\#$. It returns $\langle \mathbf{0}, \varnothing \rangle$ in the limit case $a_0 = \cdots = a_n = 0$ (lines 1 to 3); otherwise it returns representation $\langle \varphi_p, \Phi_p \rangle$ given in (13). From line 4 to line 15, the algorithm sets all P, N, α_j, β_j, and β, for $j = 0, \ldots, n$, which are used to rewrite function p in terms of p_P and p_N as in Lemma 5.1. From line 16 to line 26, it writes formulas φ_P and φ_N and adds formulas in Φ_P and Φ_N to Φ_p. For $J \in \{P, N\}$, it works throughout a $|J|$ iteration loop where each iteration takes a coefficient $\frac{a_j}{b_j}$ into account, where it treats $\frac{a_0}{b_0}$ (lines 18 to 21) separately from the others (lines 22 to 25). In lines 27 and 28 it finally writes formula φ_p and completes set Φ_p.

Theorem 5.6 *Given a rational linear function p by its coefficients, a representation $\langle \varphi_p, \Phi_p \rangle$ for $p^\#$ may be computed in polynomial time by Algorithm 3.*

Proof. Algorithm 3 builds representation $\langle \varphi_p, \Phi_p \rangle$ in (13). So, its correctness follows from Theorem 5.5. Let $[0, 1]^n$ be the domain of p and M the maximum size

Algorithm 3 REPRESENT-TL: computing representations for truncated linear functions

Input: A linear function p given by its rational coefficients $\frac{a_0}{b_0}, \frac{a_1}{b_1}, \ldots, \frac{a_n}{b_n}$.

Output: A representation $\langle \varphi_p, \Phi_p \rangle$ for the truncated function $p^{\#}$.

if $a_1 = \cdots = a_n = 0$ **then**
  return $\langle \mathbf{0}, \varnothing \rangle$;
end if
$P := \varnothing$, $N := \varnothing$;
for $j := 0, \ldots, n$ **do**
  if $a_j > 0$ **then**
    $P := P \cup \{j\}$, $\alpha_j := a_j$;
  else if $a_j < 0$ **then**
    $N := N \cup \{j\}$, $\alpha_j := -a_j$;
  end if
end for
$\beta :=$ least integer greater than or equal to $\max\{\sum_{j \in P} \frac{a_j}{b_j}, \quad -\sum_{j \in N} \frac{a_j}{b_j}\}$;
for $j \in P \cup N$ **do**
  $\beta_j := \beta \cdot b_j$;
end for
$\varphi_P := \mathbf{0}$, $\varphi_N := \mathbf{0}$, $\Phi_p := \varnothing$;
for $J = P, N$ **do**
  if $0 \in J$ **then**
    $\varphi_J := \varphi_J \oplus \text{BINARY-F}(\alpha_0, Z_{\frac{1}{\beta_0}})$;
    $\Phi_p := \Phi_p \cup \{Z_{\frac{1}{\beta_0}} \leftrightarrow \neg\text{BINARY-F}(\beta_0 - 1, Z_{\frac{1}{\beta_0}})\} \cup \text{BINARY-S}(\alpha_0, Z_{\frac{1}{\beta_0}}) \cup \text{BINARY-S}(\beta_0 - 1, Z_{\frac{1}{\beta_0}})$;
  end if
  for $j \in J \setminus \{0\}$ **do**
    $\varphi_J := \varphi_J \oplus \text{BINARY-F}(\alpha_j, Z_j^p)$;
    $\Phi_p := \Phi_p \cup \{Z_{\frac{1}{\beta_j}} \leftrightarrow \neg\text{BINARY-F}(\beta_j - 1, Z_{\frac{1}{\beta_j}}), \quad \text{BINARY-F}(\beta_j, Z_j^p) \leftrightarrow X_j, \quad Z_j^p \rightarrow Z_{\frac{1}{\beta_j}}\} \cup \text{BINARY-S}(\alpha_j, Z_j^p) \cup \text{BINARY-S}(\beta_j - 1, Z_{\frac{1}{\beta_j}}) \cup \text{BINARY-S}(\beta_j, Z_j^p)$;
  end for
end for
$\varphi_p := \text{BINARY-F}(\beta, \neg(\varphi_P \rightarrow \varphi_N))$;
$\Phi_p := \Phi_p \cup \text{BINARY-S}(\beta, \neg(\varphi_P \rightarrow \varphi_N))$;
return $\langle \varphi_p, \Phi_p \rangle$;

of a binary representation for numbers among a_j and b_j; then the input size of p is at most $2(n+1)M$. The algorithm first calculates in polynomial time all β, α_j and β_j; let μ be the maximum size of a binary representation for numbers among β, α_j and β_j. Then, it proceeds to writing the representation which is made up of at most $3(n+1)$ propositional variables of the type X_j, Z_j^p and $Z_{\frac{1}{\beta_j}}$, and $2(n+1)\mu + \mu$

propositional variables of the type ξ^k_ψ, a quantity polynomially proportional to the size of the input. Thus, the size of the representation for each propositional variable may be assumed to be a constant π also polynomially proportional to the size of the input. Next, the algorithm calculates formulas φ_P and φ_N and sets Φ_P and Φ_N in $n+1$ steps; in each one it calculates the part associated to a coefficient $\frac{\alpha_i}{\beta_i}$. For each part, computation takes polynomial time on π and at most three executions of routines BINARY-F (Algorithm 2) and BINARY-S (Algorithm 1) with argument $\langle \nu, P\rangle$, where ν is α_i, β_i or $\beta_i - 1$, which are already or may be quickly computed, and P is a propositional variable. In these cases BINARY-F and BINARY-S run in polynomial time on μ and π. The algorithm finishes calculating φ_p and Φ_p by running BINARY-F and BINARY-S with argument $\langle \beta, \neg(\varphi_P \to \varphi_N)\rangle$. Now, BINARY-F runs in polynomial time on μ and π and BINARY-S runs in polynomial time on μ, π and the size of $\neg(\varphi_P \to \varphi_N)$. After all, Algorithm 4 terminates in polynomial time. □

We call REPRESENT-TL-F and REPRESENT-TL-S the routines that separately compute φ_p and Φ_p, respectively. Both may be easily derived from routine REPRESENT-TL in Algorithm 3.

6 The General Case

Given a rational McNaughton function formatted as in Section 4, we now compute a logical representation for it. Let $f : [0,1]^n \to [0,1]$ be a rational McNaughton function in regional format with linear pieces:

$$p_i(\mathbf{x}) = \frac{a_{i0}}{b_{i0}} + \frac{a_{i1}}{b_{i1}}x_1 + \cdots + \frac{a_{in}}{b_{in}}x_n, \tag{14}$$

for $\mathbf{x} = \langle x_1, \ldots, x_n\rangle \in [0,1]^n$, $a_{ij} \in \mathbb{Z}$, $b_{ij} \in \mathbb{Z}^*_+$ and $i = 1, \ldots, m$, with each piece identical to f in region Ω_i, for $i = 1, \ldots, m$. We call ABOVE(p_k,p_i) the polynomial routine that decides if linear piece p_k is above a different linear piece p_i over Ω_i.

Let $\langle \varphi_{p_i}, \Phi_{p_i}\rangle$ be the representation for $p_i^{\#}$ given by Theorem 5.5, for $i = 1, \ldots, m$. We define:

$$\varphi = \bigvee_{i=1}^{m} \varphi_{\Omega_i}, \text{ with } \varphi_{\Omega_i} = \bigwedge_{k \in K} \varphi_{p_k}; \qquad \Phi = \bigcup_{i=1}^{m} \Phi_{p_i}; \tag{15}$$

where $k \in K$ if $p_k(\mathbf{x}) \geq p_i(\mathbf{x})$, for all $\mathbf{x} \in \Omega_i$. We are able to state the following representation result which is adapted from [17,18].

Lemma 6.1 *Let f be a rational McNaughton function in regional format with linear pieces given by (14), and let φ_{Ω_i} be a formula and Φ a set as in (15). Then, $v(\varphi_{\Omega_i}) \leq f(v(X_1), \ldots, v(X_n))$, for $v \in \mathbf{Val}_\Phi$ and $i = 1, \ldots, m$.*

Proof. Let $v \in \mathbf{Val}_\Phi$ and $\mathbf{x}_0 = \langle v(X_1), \ldots, v(X_n)\rangle$. In particular, $v \in \mathbf{Val}_{\Phi_{p_i}}$, for

φ:	$(\varphi_{p_1} \wedge \varphi_{p_2} \wedge \varphi_{p_3}) \vee (\varphi_{p_1} \wedge \varphi_{p_2} \wedge \varphi_{p_3}) \vee (\varphi_{p_1} \wedge \varphi_{p_2} \wedge \varphi_{p_3})$
Φ:	$\Phi_{p_1} \cup \Phi_{p_2} \cup \Phi_{p_3}$

Table 4
Representation as in (15) for function f from Example 4.1.

$i \in K$ and, by Theorem 5.5,

$$v(\varphi_{\Omega_i}) = \min_{k \in K} p_k^{\#}(\mathbf{x}_0).$$

If $\mathbf{x}_0 \in \Omega_i$, then $v(\varphi_{\Omega_i}) \leq p_i^{\#}(\mathbf{x}_0) = p_i(\mathbf{x}_0) = f(\mathbf{x}_0)$. On the other hand, if $\mathbf{x}_0 \notin \Omega_i$, there is some k_0 such that $\mathbf{x}_0 \in \Omega_{k_0}$. In the case $p_{k_0}(\mathbf{x}) \geq p_i(\mathbf{x})$, for all $\mathbf{x} \in \Omega_i$, then $k_0 \in K$ and $v(\varphi_{\Omega_i}) \leq p_{k_0}^{\#}(\mathbf{x}_0) = p_{k_0}(\mathbf{x}_0) = f(\mathbf{x}_0)$. In the case there is $\mathbf{x}' \in \Omega_i$ such that $p_{k_0}(\mathbf{x}') < p_i(\mathbf{x}')$, continuity of f yields that there is $t \in K$ such that $p_t(\mathbf{x}) \geq p_i(\mathbf{x})$, for all $\mathbf{x} \in \Omega_i$ and $p_t(\mathbf{x}) \leq p_{k_0}(\mathbf{x})$, for all $\mathbf{x} \in \Omega_{k_0}$. Therefore, $v(\varphi_{\Omega_i}) \leq p_t^{\#}(\mathbf{x}_0) \leq p_t(\mathbf{x}_0) \leq p_{k_0}(\mathbf{x}_0) = p_{k_0}^{\#}(\mathbf{x}_0) = f(\mathbf{x}_0)$. □

Theorem 6.2 *Any rational McNaughton function may be represented by* $\langle \varphi, \Phi \rangle$ *in* (15).

Proof. First note that any rational McNaughton function may be put in regional format as showed in Section 4. For $\langle x_1, \ldots, x_n \rangle \in [0,1]^n$, define a valuation $v \in \mathbf{Val}_\Phi$ such that $v(X_j) = x_j$ and $v(Z_j^{p_i}) = \frac{x_j}{\beta_{ij}}$, for $i = 1, \ldots, m, j = 1, \ldots, n$, $v(Z_{\frac{1}{\beta_{ij}}}) = \frac{1}{\beta_{ij}}$, for $i = 1, \ldots, m, j = 0, \ldots, n$, $v(\xi_\psi^0) = v(\psi)$, and $v(\xi_\psi^k) = \min(1, v(\xi_\psi^{k-1}) + v(\xi_\psi^{k-1}))$, for $k = 1, \ldots, \lfloor \log n \rfloor$, for any $n\psi$ that occurs in φ and Φ. Now, let $v, v' \in \mathbf{Val}_\Phi$ such that $v(X_j) = v'(X_j)$, for $j = 1, \ldots, n$. In particular, $v, v' \in \mathbf{Val}_{\Phi_{p_i}}$, for $i = 1, \ldots, m$, and, by Theorem 5.5, $v(\varphi_{p_i}) = v'(\varphi_{p_i})$, for $i = 1, \ldots, m$. Therefore, $v(\varphi) = v'(\varphi)$ and $\mathbf{X}_n$ determines φ modulo Φ-satisfiable. Finally, suppose $v \in \mathbf{Val}_\Phi$. There is some $k_0 \in K$ such that $\langle v(X_1), \ldots, v(X_n) \rangle \in \Omega_{k_0}$. Note that $v(\varphi_{\Omega_{k_0}}) = f(v(X_1), \ldots, v(X_n))$. Therefore,

$$f(v(X_1), \ldots, v(X_n)) = \max_{i=1,\ldots,m} v(\varphi_{\Omega_i}) = v(\varphi_{\Omega_{k_0}}),$$

by Lemma 6.1. □

Table 4 shows how function f in Example 4.1 can be represented as in Theorem 6.2.

Algorithm 4 returns representation $\langle \varphi, \Phi \rangle$ for function f with linear pieces given in (14). From line 1 to line 13, the algorithm writes formulas φ_{Ω_i} and the set Φ: it first computes formulas φ_{p_i} (lines 2 to 5) by means of routine REPRESENT-TL-F and then it writes φ_{Ω_i} (lines 7 to 11) by means of routine ABOVE. It writes set Φ computing each Φ_{p_i} by means of routine REPRESENT-TL-S (line 12). In line 14 it writes formula φ.

Algorithm 4 REPRESENT: computing representations for rational McNaughton functions

Input: A rational McNaughton function f in regional format given by its linear pieces coefficients $\frac{a_{10}}{b_{10}}, \ldots, \frac{a_{1n}}{b_{1n}}, \ldots, \frac{a_{m0}}{b_{m0}}, \ldots, \frac{a_{mn}}{b_{mn}}$ and regions $\Omega_1, \ldots, \Omega_m$.
Output: A representation $\langle \varphi, \Phi \rangle$ for the rational McNaughton function f.

$\Phi := \varnothing$;
for $i = 1, \ldots, m$ **do**
  $\varphi_{p_i} :=$ REPRESENT-TL-F$(\frac{a_{i0}}{b_{i0}}, \ldots, \frac{a_{in}}{b_{in}})$;
  $\varphi_{\Omega_i} := \varphi_{p_i}$;
end for
for $i = 1, \ldots, m$ **do**
  for $k = 1, \ldots, i-1, i+1, \ldots, m$ **do**
    if ABOVE$(p_k, p_i) =$ **true then**
      $\varphi_{\Omega_i} = \varphi_{\Omega_i} \wedge \varphi_{p_k}$;
    end if
  end for
  $\Phi := \Phi \cup$ REPRESENT-TL-S$(\frac{a_{i0}}{b_{i0}}, \ldots, \frac{a_{in}}{b_{in}})$;
end for
$\varphi := \varphi_{\Omega_1} \vee \cdots \vee \varphi_{\Omega_m}$;
return $\langle \varphi, \Phi \rangle$;

Theorem 6.3 *Given a rational McNaughton function f in regional format, a logical representation for it may be computed in polynomial time on the size of f by Algorithm 4.*

Proof. Algorithm 4 builds representation $\langle \varphi, \Phi \rangle$ in (15). So, the algorithm correctness follows from Theorem 6.2. The size of f is the space necessary to storage the coefficients of its m linear pieces $p_1, \ldots, p_m$ and the regions $\Omega_1, \ldots, \Omega_m$. The algorithm first calculates m representative formulas φ_{p_i} by REPRESENT-TL-F, which takes polynomial time on the size of f by Theorem 5.6. Then, it builds formulas φ_{Ω_i} from the already built representative formulas in m^2 steps; in each of these steps it runs routine ABOVE in assumed polynomial time. Along with the above computation, the algorithm also builds set Φ in m steps; in each one it calculates set Φ_{p_i} by REPRESENT-TL-S, which takes polynomial time on the size of f by Theorem 5.6. Finally, the algorithm calculates φ from formulas φ_{Ω_i} already computed. After all, Algorithm 4 terminates in polynomial time. □

7 Conclusions

We introduced a way to represent functions by logical formulas in Łukasiewicz Infinitely-valued Logic — the representation modulo satisfiability —, and we showed by a constructive proof that all rational McNaughton functions can be represented this way. Moreover, we derive an algorithm that builds such a representation in polynomial time on the size of the function. For the future, we hope to couple this algorithm with algorithms that approximate (normalized) continuous functions by

rational McNaughton functions; also, apply these approximations to the study of real systems such as neural networks through automated reasoning techniques.

References

[1] Aguzzoli, S., *The complexity of McNaughton functions of one variable*, Advances in Applied Mathematics **21** (1998), pp. 58–77.

[2] Aguzzoli, S. and D. Mundici, *Weierstrass approximations by Łukasiewicz formulas with one quantified variable*, in: *Proceedings 31st IEEE International Symposium on Multiple-Valued Logic*, IEEE, 2001, pp. 361–366.

[3] Aguzzoli, S. and D. Mundici, "Weierstrass Approximation Theorem and Łukasiewicz Formulas with one Quantified Variable," Physica-Verlag HD, Heidelberg, 2003 pp. 315–335.

[4] Amato, P., A. Di Nola and B. Gerla, *Neural networks and rational Łukasiewicz logic*, in: *2002 Annual Meeting of the North American Fuzzy Information Processing Society Proceedings. NAFIPS-FLINT 2002 (Cat. No. 02TH8622)*, IEEE, 2002, pp. 506–510.

[5] Amato, P. and M. Porto, *An algorithm for the automatic generation of a logical formula representing a control law*, Neural Network World **10** (2000), pp. 777–786.

[6] Bofill, M., F. Manya, A. Vidal and M. Villaret, *Finding hard instances of satisfiability in Łukasiewicz logics*, in: *Multiple-Valued Logic (ISMVL), 2015 IEEE International Symposium on*, IEEE, 2015, pp. 30–35.

[7] Cignoli, R., I. D'Ottaviano and D. Mundici, "Algebraic Foundations of Many-Valued Reasoning," Trends in Logic, Springer Netherlands, 2000.

[8] Esteva, F., L. Godo and F. Montagna, *The ŁΠ and ŁΠ$\frac{1}{2}$ logics: two complete fuzzy systems joining Łukasiewicz and product logics*, Archive for Mathematical Logic **40** (2001), pp. 39–67.

[9] Finger, M. and S. Preto, *Probably half true: Probabilistic satisfiability over Łukasiewicz infinitely-valued logic*, in: D. Galmiche, S. Schulz and R. Sebastiani, editors, *Automated Reasoning* (2018), pp. 194–210.

[10] Finger, M. and S. Preto, *Probably partially true: Satisfiability for Łukasiewicz infinitely-valued probabilistic logic and related topics*, Journal of Automated Reasoning (2020).
URL https://doi.org/10.1007/s10817-020-09558-9

[11] Gerla, B., *Rational Łukasiewicz logic and DMV-algebras*, Neural Network World **11** (2001), pp. 579–594.

[12] Leshno, M., V. Y. Lin, A. Pinkus and S. Schocken, *Multilayer feedforward networks with a nonpolynomial activation function can approximate any function*, Neural Networks **6** (1993), pp. 861–867.
URL www.sciencedirect.com/science/article/pii/S0893608005801315

[13] McNaughton, R., *A theorem about infinite-valued sentential logic*, Journal of Symbolic Logic **16** (1951), pp. 1–13.

[14] Mundici, D., *Satisfiability in many-valued sentential logic is NP-complete*, Theoretical Computer Science **52** (1987), pp. 145–153.

[15] Mundici, D., *A constructive proof of McNaughton's theorem in infinite-valued logic*, The Journal of Symbolic Logic **59** (1994), pp. 596–602.

[16] Rudin, W., "Principles of Mathematical Analysis," McGraw-Hill, New York, 1976, 3 edition.

[17] Tarela, J., E. Alonso and M. Martínez, *A representation method for PWL functions oriented to parallel processing*, Mathematical and Computer Modelling **13** (1990), pp. 75 – 83.
URL www.sciencedirect.com/science/article/pii/089571779090090A

[18] Tarela, J. and M. Martínez, *Region configurations for realizability of lattice piecewise-linear models*, Mathematical and Computer Modelling **30** (1999), pp. 17–27.