Mathematical Structures in Computer Science (2022), **32**, pp. 1119–1144
doi:10.1017/S096012952200010X

CAMBRIDGE UNIVERSITY PRESS

PAPER

Efficient representation of piecewise linear functions into Łukasiewicz logic modulo satisfiability†

Sandro Preto* and Marcelo Finger

Institute of Mathematics and Statistics, University of São Paulo, São Paulo, Brazil
*Corresponding author. Email: spreto@ime.usp.br

(Received 20 April 2021; revised 14 February 2022; accepted 11 April 2022; first published online 17 May 2022)

Abstract

This work concerns the representation of a class of continuous functions into Logic, so that one may automatically reason about properties of these functions using logical tools. Rational McNaughton functions may be implicitly represented by logical formulas in Łukasiewicz Infinitely-valued Logic by constraining the set of allowed valuations; such a restriction contemplates only those valuations that satisfy specific formulas. This work investigates two approaches to such depiction, called representation modulo satisfiability. Furthermore, a polynomial-time algorithm that builds this representation is presented, producing a pair of formulas consisting of the representative formula and the constraining one, given as input a rational McNaughton function in a suitable encoding. An implementation of the algorithm is discussed.

Keywords: Function representation; Łukasiewicz Infinitely-valued Logic; rational McNaughton functions; piecewise linear functions

1. Introduction

The formal representation of functions is an important step in the automated reasoning about properties of such functions; if a represented function models a system, we have at our disposal a formal way to verify desirable properties of such system by means of automated reasoning techniques. In this work, we are concerned with the theoretical and practical aspects of an efficient representation of *rational McNaughton functions*, continuous $[0, 1]$-valued piecewise linear functions with rational coefficients whose domain is $[0, 1]^n$, into Łukasiewicz Infinitely-valued Logic ($Ł_\infty$).

Rational McNaughton functions are able to approximate any continuous function $f : [0, 1]^n \to [0, 1]$ according to the Weierstrass-like result stated in Aguzzoli and Mundici (2001), Amato and Porto (2000). This property expands the application of this kind of representation practical systems modeled by any (normalized) continuous function; e.g. neural networks.

There are several approaches to represent rational McNaughton functions by logic systems; it is preferable from a computational point of view that the complexity of the target logic system

†This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. This work was carried out at the Center for Artificial Intelligence (C4AI-USP), with support by the São Paulo Research Foundation (FAPESP) (grant #2019/07665-4) and by the IBM Corporation. Sandro Preto was partly supported by the São Paulo Research Foundation (FAPESP) (grant #2021/03117-2). Marcelo Finger was partly supported by the São Paulo Research Foundation (FAPESP) (grants #2015/21880-4, #2014/12236-1); and the National Council for Scientific and Technological Development (CNPq) (grant PQ 303609/2018-4).

be as low as possible. This work aims at showing that such efficiency may be achieved by means of *representation modulo satisfiability* or *representation in the Ł$_\infty$-MODSAT system*, according to which rational McNaughton functions may be *implicitly* represented in Ł$_\infty$ (Finger and Preto, 2020). In this approach, a function is represented by a pair of Ł$_\infty$-formulas, $\langle \varphi, \Phi \rangle$, where the former is a *representative formula* and the latter a *constraining set of formulas*; a function f is represented by a pair $\langle \varphi, \Phi \rangle$, where φ is a formula that semantically acquires values $f(\mathbf{x})$, for $\mathbf{x} \in [0, 1]^n$, only from valuations in $\{v \mid v(\psi) = 1, \text{ for all } \psi \in \Phi\}$.

We introduce two different definitions for such a representation concept, which highlight different aspects of the intended technique. In a first approach, we explore the properties of logical formulas; we call it the *formula-based approach*. Such presentation differs from the original one in Finger and Preto (2020), which departs from a traditional representation and constrains the domain of the represented function; this is the *function-based approach*. We carry out a formal investigation of both views and compare them.

We present a *polynomial-time* representation builder algorithm, which is an algorithm that builds representations in Ł$_\infty$-MODSAT, taking as input rational McNaughton functions presented in *regional format*; this format has been updated in relation to a previous work as observed at the end of this section. We also discuss an implementation of such algorithm; for that, we establish classes of rational McNaughton functions with varied complexity from which random testing inputs may be generated and we present some experimental results.

The rest of this work is organized as follows. Section 2 introduces all necessary concepts of Łukasiewicz Infinitely-valued Logic and the definition of rational McNaughton function. Section 3 discusses the related work on the traditional logical representation of rational McNaughton functions. Section 4 has a theoretical investigation on the concept of representation modulo satisfiability together with two different approaches to defining such a concept and a comparison between them. Section 5 has the description of a representation builder algorithm and of an input format for such algorithm; it also discusses the input format. Section 6 presents an implementation of the algorithm, some classes of rational McNaughton functions, and experimental results. Section 7 has our conclusions.

An early conference version of this work titled *An Efficient Algorithm for Representing Piecewise Linear Functions into Logic* (Preto and Finger, 2020) was published in the proceedings of the *15th Logical and Semantic Frameworks with Applications (LSFA 2020)*. We refer the reader to that version for some omitted proofs in this work for the sake of brevity.

Note on the Previous Version. The definition of regional format that appears in Section 5.1 is modified in relation to the version in Preto and Finger (2020) and now has one additional item requiring the encoded function to obey the *lattice property*. At the time of publication of the previous version, following most current literature on lattice representation of piecewise linear functions, that property was considered a consequence of the initial definition, and it was required for the algorithm's correctness. Here we provide a counterexample, showing that such property does not follow in all cases (Example 5). Therefore, all references to the regional format in this work deal with the updated definition. Also, the input format as defined in the previous version is here called *pre-regional format* and it is investigated in Section 5.4 together with a literature review. The aforementioned omitted proofs are not affected by the update in the definition of regional format encoding of rational McNaughton functions.

2. Preliminaries

Łukasiewicz Infinitely-valued Logic (Ł$_\infty$) is arguably one of the best studied many-valued logics (Cignoli et al., 2000), whose satisfiability over Ł$_\infty$ is NP-complete (Mundici, 1987), so it is in the same complexity class as classical propositional satisfiability (SAT); there are a number of available implementations of Ł$_\infty$-solvers, for which an empirical phase transition phenomenon is identified (Bofill et al., 2015; Finger and Preto, 2018).

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

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

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

$$v(\neg\varphi) = 1 - v(\varphi). \tag{2}$$

Consider a function $v_{\mathbb{P}}$ that maps propositional variables to a value in the interval $[0, 1]$ and extend it to a valuation by obeying (1) and (2). This extension is uniquely defined, and also called $v_{\mathbb{P}}$.

Let **Val** be the set of all valuations; let $\mathrm{Var}(\Phi)$ be the set of all propositional variables occurring in the formulas $\varphi \in \Phi$; and let $\mathbf{X}_n$ be the set of propositional variables $\{X_1, \ldots, X_n\} \subset \mathbb{P}$. A formula φ is *satisfiable* (or 1-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 satisfy a set of formulas Φ; we call such a restricted set of valuations a *semantics modulo satisfiability*.

From disjunction and negation, we derive the following operators:

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

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 $0\varphi =_{\mathrm{def}} \mathbf{0}$ and $n\varphi =_{\mathrm{def}} \varphi \oplus \cdots \oplus \varphi$, n times, for $n \in \mathbb{N}^*$; and $\bigoplus_{i \in \emptyset} \varphi_i =_{\mathrm{def}} \mathbf{0}$.

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 $[0, 1]$ real number interval;
- 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.

A *McNaughton function* is a particular case of rational McNaughton function whose linear pieces coefficients are restricted to integer values.

Let $\Omega \subset [0, 1]^n$; we denote by Ω° its interior, by $\mathrm{cl}(\Omega)$ its closure, by $\partial\Omega$ its boundary and by $\mathrm{conv}(\Omega)$ its convex hull.

3. The Traditional Way and Related Work

In the traditional way of representing functions by logical formulas, we inductively associate to a given formula φ, with $\mathrm{Var}(\varphi) \subset \mathbf{X}_n$, 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) $f_{\varphi\oplus\psi}(x_1, \ldots, x_n) = \min(1, f_\varphi(x_1, \ldots, x_n) + f_\psi(x_1, \ldots, x_n))$.

Note that the definition of f_φ depends on n. It follows that f_φ enjoys the property:

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

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

And, then, we say that formula φ *represents* function f. For any formula φ, f_φ is a McNaughton function and reciprocally, McNaughton's Theorem states that every McNaughton function f may be represented by some formula φ (McNaughton, 1951; Mundici, 1994).

There are propositional logics whose formulas represent rational McNaughton functions in the traditional way. In the following, we present and discuss some of such most relevant approaches.

- Logic $\text{Ł}\Pi\frac{1}{2}$ extends Ł_∞ with a product operator, its residuum and a constant expressing the truth value $\frac{1}{2}$, not directly expressible in Ł_∞ (Esteva et al., 2001). That logic not only allows for the expressivity of rational McNaughton functions but also expresses piecewise polynomials; as a consequence satisfiability over $\text{Ł}\Pi\frac{1}{2}$ requires finding roots of polynomials of n-degree making its complexity extremely high.
- Logic $\exists\text{Ł}$ also expresses rational McNaughton functions (Aguzzoli and Mundici, 2001, 2003); it extends Ł_∞ and introduces rational numbers by providing restricted form of propositional quantification whose semantic counterpart is the maximization of a set of Ł_∞-valuations of a formula. Satisfiability problem in that logic is in the complexity class Σ_2^p, which is also a high complexity.
- Rational Łukasiewicz Logic extends Ł_∞ with division operators δ_n that induces division by $n \in \mathbb{N}^*$ in its semantics, i.e. $v(\delta_n\varphi) = \frac{v(\varphi)}{n}$, where v is a valuation of Rational Łukasiewicz Logic (Gerla, 2001); its associated tautology problem is coNP-complete, which is a reasonable complexity for this task. This logic expresses all rational McNaughton functions, but no algorithm has ever been proposed that builds the representation formulas; an attempt to derive such algorithm from the results of Gerla (2001) would lead to the problem of representing McNaughton functions in Ł_∞; it is known that this task may be done in polynomial time on the coefficients of some specific functions (Aguzzoli, 1998), however these methods lead to exponential time complexity if binary representation of the coefficients is used.
- Logic RŁ extends Ł_∞ with constant multiplication operators ∇_r that induces multiplication by $r \in [0, 1]$ in its semantics, i.e. $v(\nabla_r\varphi) = r \cdot v(\varphi)$, where v is a RŁ-valuation (Di Nola and Leuștean, 2011, 2014). This logic expresses all continuous $[0, 1]$-valued piecewise linear functions over $[0, 1]^n$; in particular, it expresses all rational McNaughton functions, however its language is uncountable, thus it is not computable. We are unaware of computational considerations so far about the fragment of RŁ that comprehends only operators ∇_q, for $q \in [0, 1] \cap \mathbb{Q}$.

4. Representation Modulo Satisfiability

Although formulas of Ł_∞ only represent (integer) McNaughton functions, we present here an implicit representation of rational McNaughton functions, which we call *representation modulo satisfiability*. Next, we introduce two different definitions for such a representation concept, which highlight distinct aspects of the intended technique, then we compare both approaches.

4.1 The formula-based approach

We start by analyzing the property which is a crux for the possibility that logical formulas represent functions in the traditional way: the 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 $\text{Var}(\varphi) \subset \mathbf{X}$. Thus, if $\mathbf{X}$ is semantically identified to the domain of a function, formula φ may semantically express all the values such function. Let us generalize this notion.

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

Definition 1. *Let φ be a formula and let Φ be a set of formulas. We say that a set of propositional 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 $j = 1, \ldots, n$; and*
- *For any pair of valuations $v, v' \in \mathbf{Val}_\Phi$ such that $v(X_j) = v'(X_j)$, for $j = 1, \ldots, n$, we have that $v(\varphi) = v'(\varphi)$—i.e. valuations in $\mathbf{Val}_\Phi$ are truth-functional on variables in $\mathbf{X}_n$.*

For instance, for any formula φ such that $\mathrm{Var}(\varphi) \subset \mathbf{X}_n$, $\mathbf{X}_n$ determines φ modulo $\emptyset$-satisfiable, by truth-functionality and the fact that $\mathbf{Val}_\emptyset = \mathbf{Val}$. Then, representation modulo satisfiability in the formula-based approach is defined in a way that retrieves property (3).

Definition 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; and*
- *$f(v(X_1), \ldots, v(X_n)) = v(\varphi)$, for all $v \in \mathbf{Val}_\Phi$.*

The definition of representation modulo satisfiability in the formula-based approach adapts property (3) from unrestricted to restricted valuations, as required by the semantics modulo satisfiability. The next example should clarify the usage of such property.

Example 1. The function $f : [0,1] \to [0,1]$, given by $f(x_1) = \frac{x_1}{2}$, may be represented by $\langle Z_1, \Phi\rangle$, where $\Phi = \{Z_1 \oplus Z_1 \leftrightarrow X_1,\ Z_{1/2} \leftrightarrow \neg Z_{1/2},\ Z_1 \to Z_{1/2}\}$. Propositional variable X_1 is intended to take values in the domain of function f and Z_1 is intended to take half the value of X_1; it is also necessary to define constant $\frac{1}{2}$ by propositional variable $Z_{1/2}$ and assure Z_1 takes at most value $\frac{1}{2}$. Observe that X_1 determines $\varphi = Z_1$ modulo Φ-satisfiable since, if one associates a value x_1 in the domain of function f to propositional variable X_1—making $x_1 = v(X_1)$—, the value $v(Z_1)$ of formula Z_1 is uniquely determined modulo satisfiability of Φ, i.e. assuming that valuation v satisfies Φ. Moreover, the pair $\langle Z_1, \Phi\rangle$ represents function f, since $f(x_1) = v(Z_1)$.

4.2 The function-based approach

Finger and Preto (2020) proposed a function-based definition of representation modulo satisfiability that constained the domain of a traditionally represented function. We review such approach in the following.

We extend the notion of associating functions from formulas to a pair $\langle \varphi, \Phi\rangle$, where φ is a formula and Φ is a set of formulas, with $\mathrm{Var}(\varphi) \cup \mathrm{Var}(\Phi) \subset \mathbf{X}_m$, as follows. First, let the function domain be

$$D_{\langle \varphi, \Phi\rangle} = \left\{ \langle x_1, \ldots, x_m\rangle \in [0,1]^m \;\middle|\; f_\psi(x_1, \ldots, x_m) = 1, \text{ for all } \psi \in \Phi \right\}.$$

Then we inductively define function $f_{\langle \varphi, \Phi\rangle} : D_{\langle \varphi, \Phi\rangle} \to [0,1]$ by the following clauses in total analogy to (i)–(iii) in the beginning of Section 3:

(i) $f_{\langle X_j, \Phi\rangle}(x_1, \ldots, x_m) = x_j$, for $j = 1, \ldots, m$;

(ii) $f_{\langle \neg\varphi, \Phi\rangle}(x_1, \ldots, x_m) = 1 - f_{\langle \varphi, \Phi\rangle}(x_1, \ldots, x_m)$;

(iii) $f_{\langle \varphi \oplus \psi, \Phi\rangle}(x_1, \ldots, x_m) = \min\big(1, f_{\langle \varphi, \Phi\rangle}(x_1, \ldots, x_m) + f_{\langle \psi, \Phi\rangle}(x_1, \ldots, x_m)\big)$.

The definitions of $D_{\langle \varphi, \Phi\rangle}$ and $f_{\langle \varphi, \Phi\rangle}$ depend on m. In the function-based approach, we have the following definition.

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

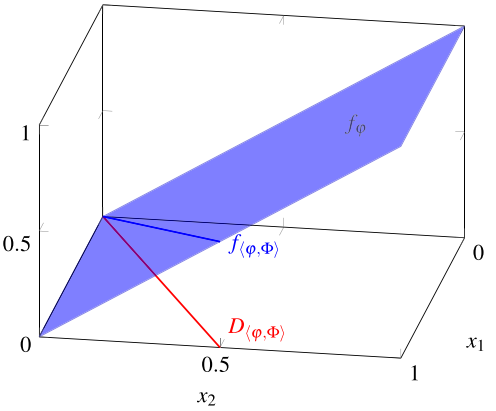


Figure 1. Graphs of functions f_φ and $f_{\langle\varphi,\Phi\rangle}$ and of set $D_{\langle\varphi,\Phi\rangle}$ in Example 2, for fixed $x_3 = \frac{1}{2}$.

Definition 3. *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* functionally represents *f* modulo Φ-satisfiable *or that $\langle\varphi, \Phi\rangle$* functionally represents *f* (in the system Ł$_\infty$-MODSAT) *if $Var(\varphi) \cup Var(\Phi) = \mathbf{X}_m$, $m \geq n$, and there exist $m - n$ functions $z_j : [0,1]^n \to [0,1]$, $j = 1, \ldots, m-n$, such that:*

- *For any $\langle x_1, \ldots, x_m\rangle \in D_{\langle\varphi,\Phi\rangle}$, $x_{n+j} = z_j(x_1, \ldots, x_n)$, $j = 1, \ldots, m-n$;*
- *For any $\langle x_1, \ldots, x_n\rangle \in [0,1]^n$,*
  $f(x_1, \ldots, x_n) = f_{\langle\varphi,\Phi\rangle}(x_1, \ldots, x_n, z_1(x_1, \ldots, x_n), \ldots, z_{m-n}(x_1, \ldots, x_n))$.

We write $\mathbf{x} = \langle x_1, \ldots, x_n\rangle$ and $\mathbf{z} = \langle x_{n+1}, \ldots, x_m\rangle$.

In the functional representation modulo satisfiability, a pair $\langle\varphi, \Phi\rangle$ functionally represents a function $f : [0,1]^n \to [0,1]$ when formula φ is the traditional representation of another function $f_\varphi : [0,1]^m \to [0,1]$ whose domain $[0,1]^m$ has possibly higher dimension – $m \geq n$ – and can be constrained to $D_{\langle\varphi,\Phi\rangle} \subset [0,1]^m$ in order to be identified with the domain $[0,1]^n$ of the original function f; elements $\mathbf{x} \in [0,1]^n$ are identified with elements $\langle\mathbf{x}, \mathbf{z}\rangle \in D_{\langle\varphi,\Phi\rangle}$ and it must hold that $f(\mathbf{x}) = f_{\langle\varphi,\Phi\rangle}(\mathbf{x}, \mathbf{z})$. Note that the constraining from $[0,1]^m$ to $D_{\langle\varphi,\Phi\rangle}$ is a disguised application of semantics modulo satisfiability since $\langle x_1, \ldots, x_m\rangle \in D_{\langle\varphi,\Phi\rangle}$ if, and only if, there is a valuation $v \in \mathbf{Val}_\Phi$ such that $v(X_1) = x_1, \ldots, v(X_m) = x_m$.

Example 2. The representation for function $f : [0,1] \to [0,1]$, given by $f(x_1) = \frac{x_1}{2}$, in Example 1 is almost a functional representation for it; we only need to replace Z_1 and $Z_{1/2}$ by X_2 and X_3 to fit the definition, which results in $\langle X_2, \Phi\rangle$, where $\Phi = \{X_2 \oplus X_2 \leftrightarrow X_1,\ X_3 \leftrightarrow \neg X_3,\ X_2 \to X_3\}$. In this case, we have $n = 1$, $m = 3$ and

$$D_{\langle\varphi,\Phi\rangle} = \left\{ \langle x_1, x_2, x_3\rangle \in [0,1]^3 \;\middle|\; x_1 \in [0,1],\ \ x_2 = \frac{x_1}{2},\ \ x_3 = \frac{1}{2} \right\}.$$

Then, there are functions $z_1 : [0,1] \to [0,1]$ and $z_2 : [0,1] \to [0,1]$ given by $z_1(x_1) = \frac{x_1}{2}$ and $z_2(x_1) = \frac{1}{2}$. And we have that

$$f(x_1) = \frac{x_1}{2} = z_1(x_1) = f_{\langle\varphi,\Phi\rangle}(x_1, z_1(x_1), z_2(x_1)).$$

Note that function $f_\varphi : [0,1]^3 \to [0,1]$ is given by $f_\varphi(x_1, x_2, x_3) = x_2$. Figure 1 has graphs of the functions f_φ and $f_{\langle\varphi,\Phi\rangle}$ and of the set $D_{\langle\varphi,\Phi\rangle}$.

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

4.3 Formula-based versus function-based approaches

We have seen two attempts to formalize a concept of representation modulo satisfiability. Each approach has the virtue of elucidating some different aspects of the technique that provides a pair $\langle \varphi, \Phi \rangle$, where φ is a representative formula and Φ is a set of constraining formulas. In this way, one might wonder whether they formalize the same concept. In fact, despite the similarity, the function-based presentation is a bit more restrictive than the formula-based one; the former constrains the values of x_{n+j}, for $j = 1, \ldots, m-n$, to be functions $z_j(x_1, \ldots, x_n)$, for any $\langle x_1, \ldots, x_m \rangle \in D_{\langle \varphi, \Phi \rangle}$, so that the set $D_{\langle \varphi, \Phi \rangle}$ is minimal.

Example 3. The pair $\langle X_2 \oplus X_3, \Phi \rangle$, where $\Phi = \{X_2 \leftrightarrow \neg X_3\}$, is a representation in the formula-based approach for the constant function $f : [0,1] \to [0,1]$, given by $f(x_1) = 1$, but not in the function-based approach, as the representation is not functional. In fact, for a given $\alpha \in [0,1]$, it is not possible to determine a unique element $\langle x_1, x_2, x_3 \rangle \in D_{\langle \varphi, \Phi \rangle}$ such that $x_1 = \alpha$, since $\langle \alpha, \beta, 1-\beta \rangle \in D_{\langle \varphi, \Phi \rangle}$, for any $\beta \in [0,1]$.

In order to make both presentations equivalent, we should restrict the formula-based one by adding to Definition 1 the following items:

- $\mathrm{Var}(\varphi) \cup \mathrm{Var}(\Phi) = \mathbf{X}_m$, $m \geq n$;
- For any pair of valuations $\nu, \nu' \in \mathbf{Val}_\Phi$ such that $\nu(X_j) = \nu'(X_j)$, for $j = 1, \ldots, n$, we have that $\nu(X_j) = \nu'(X_j)$, for $j = n+1, \ldots, m$.

The first item above only standardizes propositional variables to appear in φ and Φ; such standardization was intended to ease the inductive process of associating functions to formulas in the formula-based approach. The second item constrains, for valuations in $\mathbf{Val}_\Phi$, the values of propositional variables in $\mathbf{X}_m \setminus \mathbf{X}_n$ as functions of the values of propositional variables in $\mathbf{X}_n$; this is stronger than the original definition which only constrains the value of φ – indeed these new items yield that the value of φ is invariant – and is the counterpart to the constraints to elements in $D_{\langle \varphi, \Phi \rangle}$ by functions $z_1, \ldots, z_{m-n}$. We refer to the more restrictive form of Definition 1 as *strong determination modulo satisfiability* and to the consequent more restrictive form of Definition 2 as *strong representation modulo satisfiability*.

Theorem 4. *A pair $\langle \varphi, \Phi \rangle$ strongly represents a function $f : [0,1]^n \to [0,1]$ (in the formula-based approach) if, and only if, it functionally represents f (in the function-based approach).*

Proof. Let $\langle \varphi, \Phi \rangle$ be a strong representation for f (in the formula-based approach); then $\mathrm{Var}(\varphi) \cup \mathrm{Var}(\Phi) = \mathbf{X}_m$. For any $\mathbf{x} = \langle x_1, \ldots, x_n \rangle \in [0,1]^n$ and $j = 1, \ldots, m-n$, we set $z_j(\mathbf{x}) = \nu_{\mathbf{x}}(X_{n+j})$, where $\nu_{\mathbf{x}} \in \mathbf{Val}_\Phi$ is such $\nu_{\mathbf{x}}(X_j) = x_j$, for $j = 1, \ldots, n$. This way, for any $\langle x_1, \ldots, x_m \rangle \in D_{\langle \varphi, \Phi \rangle}$, $\psi \in \Phi$ and valuation ν, with $\nu(X_j) = x_j$, for $j = 1, \ldots, m$, we have that $\nu(\psi) = f_\psi(x_1, \ldots, x_m) = 1$; then $\nu \in \mathbf{Val}_\Phi$ and $x_{n+j} = \nu(X_{n+j}) = \nu_{\langle x_1, \ldots, x_n \rangle}(X_{n+j}) = z_j(x_1, \ldots, x_n)$, for $j = 1, \ldots, m-n$. Finally, for any $\langle x_1, \ldots, x_n \rangle \in [0,1]^n$, there is a $\nu \in \mathbf{Val}_\Phi$ such that $\nu(X_i) = x_i$, for $i = 1, \ldots, n$. Therefore, $\langle \nu(X_1), \ldots, \nu(X_m) \rangle \in D_{\langle \varphi, \Phi \rangle}$, $f(x_1, \ldots, x_n) = f(\nu(X_1), \ldots, \nu(X_n)) = \nu(\varphi) = f_{\langle \varphi, \Phi \rangle}(\nu(X_1), \ldots, \nu(X_m)) = f(x_1, \ldots, x_n, z_1(x_1, \ldots, x_n), \ldots, z_{m-n}(x_1, \ldots, x_n))$ and $\langle \varphi, \Phi \rangle$ is a functional representation for f (in the function-based approach). Conversely, let $\langle \varphi, \Phi \rangle$ be a functional representation for f (in the function-based approach); then $\mathrm{Var}(\varphi) \cup \mathrm{Var}(\Phi) = \mathbf{X}_m$. Since for any $\mathbf{x} = \langle x_1, \ldots, x_n \rangle \in [0,1]^n$ there are values $z_j(\mathbf{x})$, $j = 1, \ldots, m-n$, such that $\langle \mathbf{x}, z_1(\mathbf{x}), \ldots, z_{m-n}(\mathbf{x}) \rangle \in D_{\langle \varphi, \Phi \rangle}$, then, for all $\psi \in \Phi$, $\nu(\psi) = f_\psi(\mathbf{x}, z_1(\mathbf{x}), \ldots, z_{m-n}(\mathbf{x})) = 1$, for a valuation $\nu \in \mathbf{Val}_\Phi$, where $\nu(X_1) = x_1, \ldots, \nu(X_n) = x_n, \nu(X_{n+1}) = z_1(\mathbf{x}), \ldots, \nu(X_m) = z_{m-n}(\mathbf{x})$. For $\nu, \nu' \in \mathbf{Val}_\Phi$, where $\mathbf{x} = \langle \nu(X_1), \ldots, \nu(X_n) \rangle = \langle \nu'(X_1), \ldots, \nu'(X_n) \rangle$, we have $\langle \mathbf{x}, \nu(X_{n+1}), \ldots, \nu(X_m) \rangle, \langle \mathbf{x}, \nu'(X_{n+1}), \ldots, \nu'(X_m) \rangle \in D_{\langle \varphi, \Phi \rangle}$, then $\nu(X_{n+j}) = z_j(\mathbf{x}) = \nu'(X_{n+j})$,

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

for $j = 1, \ldots, m-n$, and $v(\varphi) = f_{\langle\varphi,\Phi\rangle}(\mathbf{x}, z_1(\mathbf{x}), \ldots, z_{m-n}(\mathbf{x})) = v'(\varphi)$; therefore, $\mathbf{X}_n$ strongly determines φ modulo Φ-satisfiable. Also, $f(\mathbf{x}) = f_{\langle\varphi,\Phi\rangle}(\mathbf{x}, z_1(\mathbf{x}), \ldots, z_{m-n}(\mathbf{x})) = v(\varphi)$, for $v \in \mathbf{Val}_\Phi$, and $\langle\varphi, \Phi\rangle$ is a strong representation for f (in the formula-based approach). □

From now on, we choose to deal with the formula-based approach to represent functions modulo satisfiability, as we deem it a clearer and less restrictive definition. Moreover, we will only refer to the version presented in Section 4.1 (not the strong one) as fixing the value of φ modulo Φ-satisfiable is enough for establishing a satisfactory concept of representation. However, all the constructions of representations to follow also fix the values of the additional propositional variables (other than the ones in $\mathbf{X}_n$) modulo Φ-satisfiable; thus, for them to be strong representations, only the standardization of the propositional variables is missing.

5. An Efficient Algorithm for Building Representations

An attempt to derive a representation algorithm from the representation results in Finger and Preto (2020) would lead to an exponential explosion, since the proposed pairs $\langle\varphi, \Phi\rangle$ for representing only truncated linear functions are already exponential over the binary representation of their coefficients. Thus, we need to produce a less complex representation in order to derive an efficient algorithm that actually represents a piecewise linear function; this is our aim in this section.

5.1 Regional format of rational McNaughton functions

The algorithm we present later uses a lattice representation of rational McNaughton functions; before that we employ an encoding as follows. A rational McNaughton function $f : [0, 1]^n \to [0, 1]$ is in *regional format* if it 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:

- $\bigcup_{i=1}^m \Omega_i = [0, 1]^n$;
- $\Omega_i^\circ \cap \Omega_j^\circ = \emptyset$, for $i \neq j$;
- 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 $\mathbf{x} \in \Omega_i$;
- Linear pieces and regions satisfy the *lattice property*, that is, for $i \neq j$, there is k such that linear piece p_i is above linear piece p_k over region Ω_i and linear piece p_k is above linear piece p_j over region Ω_j.

The regional format allows for the repetition of linear pieces so that there is a one-to-one correspondence between regions and pieces. In this format, 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 encoding. We discuss the regional format further at the end of this section.

Example 4. Rational McNaughton function f with graph in Figure 2a may be given by the linear pieces $p_1(x_1, x_2) = \frac{4}{9} + \frac{2}{3}x_2$, $p_2(x_1, x_2) = \frac{5}{6} - \frac{1}{2}x_2$ and $p_3(x_1, x_2) = \frac{4}{3} - x_1$. Regions Ω_i associated to each linear piece are depicted in Figure 2b and described in Table 1; we soon tackle the problem of deciding if a linear piece is above another.

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

Table 1. Regions Ω_i for function f in Example 4

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

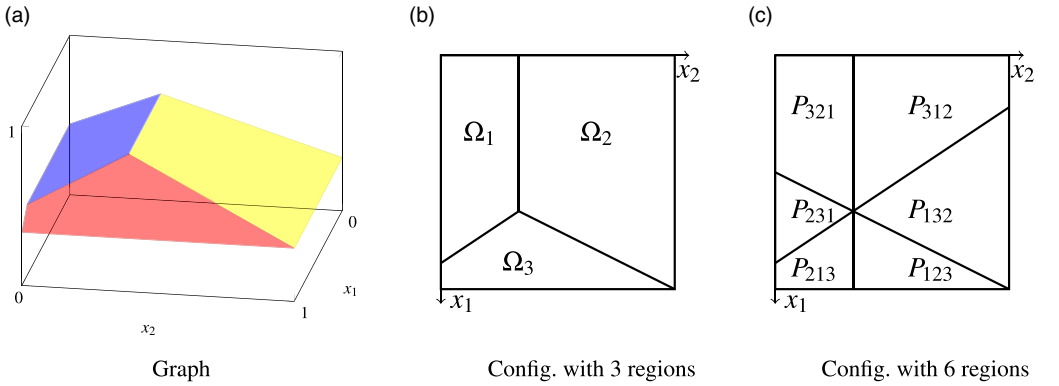


Figure 2. Graph of rational McNaughton function in Example 4.

Let us deal with the encoding of regions. First, we characterize them in next result.

Lemma 5. *Closures of regions in regional format of rational McNaughton functions are polyhedra.*

Proof. Let Ω be a region of a rational McNaughton function in regional format. Since $\mathrm{cl}(\Omega)$ is a convex compact set, it is the convex hull of its extreme points. Suppose $\mathrm{cl}(\Omega)$ has infinitely many extreme points and let E be the set comprehending the infinitely many extreme points which are in the interior of $[0,1]^n$. Let $U = \bigcup\{\Omega_i \mid \Omega_i \neq \Omega\}$; we have that $E \subset \partial U$ and, since U is a finite union of regions Ω_i, there exists an infinite set $E' \subset E$, such that $E' \subset \partial\Omega' \subset \mathrm{cl}(\Omega')$, for some $\Omega' = \Omega_i \neq \Omega$. Let $E_{n+1} \subset E'$ be a set with $n+1$ points; as $\mathrm{cl}(\Omega)$ and $\mathrm{cl}(\Omega')$ are convex sets, we have that $\mathrm{conv}(E_{n+1}) \subset \mathrm{cl}(\Omega) \cap \mathrm{cl}(\Omega')$. Also, as Ω and Ω' are convex sets, we have that $\Omega^\circ = \mathrm{cl}(\Omega)^\circ$ and $\Omega'^\circ = \mathrm{cl}(\Omega')^\circ$. Finally, since $\mathrm{conv}(E_{n+1})$ is an n-simplex, it follows that $\Omega^\circ \cap \Omega'^\circ \neq \emptyset$, contrary to the definition of regional format. Therefore, $\mathrm{cl}(\Omega)$ is a polyhedron. □

Since the closure $\mathrm{cl}(\Omega)$ of region Ω is a polyhedron, it may be entirely described as the finite intersection of half-spaces given by linear inequalities as

$$\mathrm{cl}(\Omega) = \left\{ \mathbf{x} \in [0,1]^n \;\middle|\; \omega_{i0} + \omega_{i1}x_1 + \cdots + \omega_{in}x_n \geq 0,\ i = 1, \ldots, \lambda_\Omega \right\}. \tag{5}$$

We show a polynomial procedure for deciding if a linear piece p_k is above another linear piece p_i over region Ω_i that takes polyhedron $\mathrm{cl}(\Omega_i)$ given by (5) as input. Let p_k and p_i be given by

$$p_k(\mathbf{x}) = \gamma_{k0} + \gamma_{k1}x_1 + \cdots + \gamma_{kn}x_n,$$
$$p_i(\mathbf{x}) = \gamma_{i0} + \gamma_{i1}x_1 + \cdots + \gamma_{in}x_n,$$

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

Algorithm 1 ABOVE-MAX: decides if a linear piece is above another one over a region

Input: Linear pieces p_k and p_i given by their coefficients $\gamma_{k0}, \ldots, \gamma_{kn}, \ldots, \gamma_{i0}, \ldots, \gamma_{in}$ and polyhedron $\mathrm{cl}(\Omega_i)$.
Output: **True**, if p_k is above p_i over Ω_i. Or **False**, otherwise.

```
M := MAX(p_i − p_k, cl(Ω_i));
if M ≤ 0 then
    return True;
else
    return False;
end if
```

where $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$. In order to decide if p_k is above p_i over Ω_i, Algorithm 1 analyzes the optimal value of the maximization linear program:

$$\begin{aligned} &\max && p_i - p_k \\ &\text{subject to} && \mathrm{cl}(\Omega_i) \end{aligned}$$

We call $\mathrm{MAX}(f, P)$ the routine that computes maximum value of an objective function f over polyhedron P. It is known that such linear programming problem may be solved in polynomial time (Bertsimas and Tsitsiklis, 1997).

Theorem 6. *Given linear pieces p_k and p_i and a polyhedron $cl(\Omega_i)$, Algorithm 1 decides in polynomial time whether or not p_k is above p_i over Ω_i.*

Proof. We have that $p_k(\mathbf{x}) \geq p_i(\mathbf{x})$, for $\mathbf{x} \in \Omega_i$ if, and only if,

$$p_i(\mathbf{x}) - p_k(\mathbf{x}) \leq 0, \tag{6}$$

for $\mathbf{x} \in \Omega_i$. Let M be the maximum value of the objective function $p_i(\mathbf{x}) - p_k(\mathbf{x})$ in $\mathrm{cl}(\Omega_i)$ and let $\mathbf{x}_M \in \mathrm{cl}(\Omega_i)$ be an argument where the objective function has value M. In case $M \leq 0$, then (6) is satisfied by all $\mathbf{x} \in \Omega_i \subset \mathrm{cl}(\Omega_i)$. In case $M > 0$, then, either $\mathbf{x}_M \in \Omega_i$ fails to fulfill (6) or, if $\mathbf{x}_M \in \partial\Omega_i$, there is some $\mathbf{x} \in \Omega_i$ which fails to do so, by the continuity of the objective function. The correctness of Algorithm 1 follows from these remarks and, as MAX is a polynomial routine, it terminates in polynomial time. □

In view of Theorem 6, it is enough to encode regions Ω in such a way that there is a polynomial procedure to compute $\mathrm{cl}(\Omega)$ as in (5). Moreover, from continuity of rational McNaughton function f, we have that $f(\mathbf{x}) = p_i(\mathbf{x})$, for any $\mathbf{x} \in \mathrm{cl}(\Omega_i)$, so a natural standardization is to consider regions that are already polyhedra given by (5). We say that functions given this way are in *closed regional format*; this is the case in Example 4.

We should establish that any rational McNaughton function may be put in (closed) regional format. Let $f : [0,1]^n \to [0,1]$ be a rational McNaughton function with distinct linear pieces $p_1, \ldots, p_{\bar{m}}$; for each permutation ρ of the set $\{1, \ldots, \bar{m}\}$, we define the polyhedron

$$P_\rho = \left\{ \mathbf{x} \in [0,1]^n \;\middle|\; p_{\rho(1)}(\mathbf{x}) \geq \cdots \geq p_{\rho(\bar{m})}(\mathbf{x}) \right\}. \tag{7}$$

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

Theorem 7. *The set $\mathscr{C}$ has the following properties.*

(1) $\bigcup \mathscr{C} = [0,1]^n$.
(2) For polyhedron $P \in \mathscr{C}$ and indexes $i', i'' \in \{1, \ldots, \bar{m}\}$ with $i' \neq i''$, $p_{i'}(\mathbf{x}) \neq p_{i''}(\mathbf{x})$, for any $\mathbf{x} \in P^\circ$.

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

(3) $P'^{\circ} \cap P''^{\circ} = \emptyset$, *for* $P', P'' \in \mathscr{C}$ *such that* $P' \neq P''$.
(4) *For each polyhedron* $P \in \mathscr{C}$, *there is an index* $i_P \in \{1, \ldots, \bar{m}\}$ *such that* $f(\mathbf{x}) = p_{i_P}(\mathbf{x})$, *for* $\mathbf{x} \in P$.
(5) *For polyhedra* $P', P'' \in \mathscr{C}$, *there is an index* $k \in \{1, \ldots, \bar{m}\}$ *such that* $p_{i_{P'}}$ *is above* p_k *over* P' *and* p_k *is above* $p_{i_{P''}}$ *over* P''.

Proof. For (1)–(4), see Preto and Finger (2020, Proposition 4.2); the proof of (5), based on the proof in Cignoli et al. (2000, Proposition 9.1.4), follows. If $p_{i_{P'}}$ is not above $p_{i_{P''}}$ over P'', there is $\mathbf{b} \in P''^{\circ}$ such that $p_{i_{P'}}(\mathbf{b}) \leq p_{i_{P''}}(\mathbf{b})$. Let $\mathbf{a} \in P'^{\circ}$ and $\mathbf{A}, \mathbf{B} \in [0,1]^{n+1}$ be such that $\mathbf{A} = \langle \mathbf{a}, f(\mathbf{a}) \rangle$ and $\mathbf{B} = \langle \mathbf{b}, f(\mathbf{b}) \rangle$; also, let g be the restriction of f to the line segment $[\mathbf{a}, \mathbf{b}] = \{(1-\lambda)\mathbf{a} + \lambda\mathbf{b} \mid \lambda \in [0,1]\}$. There is a point $\mathbf{a}' \in [\mathbf{a}, \mathbf{b}] \setminus \{\mathbf{a}\}$, such that g coincides with $p_{i_{P'}}$ over $[\mathbf{a}, \mathbf{a}']$; since the graph of g lies strictly below $[\mathbf{A}, \mathbf{B}]$ over $[\mathbf{a}, \mathbf{a}'] \setminus \{\mathbf{a}\}$, among all $\mathbf{c} \in [\mathbf{a}, \mathbf{b}] \setminus \{\mathbf{a}\}$ such that $\langle \mathbf{c}, g(\mathbf{c}) \rangle \in [\mathbf{A}, \mathbf{B}]$, there is one point $\mathbf{d}$ nearest to $\mathbf{a}$ (possibly $\mathbf{b}$). Let $k \in \{1, \ldots, \bar{m}\}$ be such that $g(\mathbf{d}) = p_k(\mathbf{d})$ and g coincides with p_k on a nonempty line segment $[\mathbf{d}', \mathbf{d}] \subset [\mathbf{a}, \mathbf{d}]$; the restriction of the graph of p_k to $[\mathbf{d}', \mathbf{d}] \setminus \{\mathbf{d}\}$ must be strictly below $[\mathbf{A}, \mathbf{B}]$. Then, $p_k(\mathbf{a}) < p_{i_{P'}}(\mathbf{a})$, which makes $p_{i_{P'}}$ to be above p_k over P'. We also have that $p_{i_{P''}}(\mathbf{b}) < p_k(\mathbf{b})$, which makes p_k to be above $p_{i_{P''}}$ over P''. □

Polyhedra in $\mathscr{C}$ may play the role of regions in regional format since they are convex sets with the properties above; note that the same linear piece p_i may be f-associated to many distinct polyhedra. Determining whether a linear piece p_k is above other linear piece p_i over $P \in \mathscr{C}$ boils down to comparing their values for some point $\mathbf{x} \in P^{\circ}$. Thus, any rational McNaughton function may be encoded in regional format. Figure 2c shows the permutation-based region configuration $\mathscr{C}$ for the function in Example 4.

The regional format assures sufficient conditions and information about the ordering of linear pieces over its region configuration which are required for a lattice representation, i.e. a representation that comes from the application of lattice operations to the linear pieces of a given continuous piecewise linear function. For instance, Mundici (1994) uses a lattice representation for representing McNaughton functions in Ł$_\infty$– which was adapted by Finger and Preto (2020) for representing rational McNaughton functions in Ł$_\infty$-MODSAT – that requires conditions and information from the region configuration given by the decomposition in simplices of the polyhedra P_ρ in $\mathscr{C}$; this path has also been followed in the literature for representing continuous piecewise linear functions in other Ł$_\infty$-based logical systems; see Section 3.

The setback with describing a rational McNaughton function using the set $\mathscr{C}$ of polyhedra is that, in the worst case, $|\mathscr{C}| = \bar{m}!$; the situation may be even worse when decompositions in simplices are considered. However, in many cases, regional format is able to encompass sufficient conditions and information for lattice representation with a smaller set of regions; Figure 2 shows such contrast related to Example 4 and it may also be seen in the classes of functions in Section 6. This feature does not interfere with the complexity of the general algorithm in Section 5.3, since it only amounts to a possible reduction of the input size, but it might yield a gain in the complexity of the representation of inputs and, therefore, make some applications viable. Of course that if a more compact encoding of rational McNaughton functions is provided, a side effect might be an inefficient translation from such encoding to the regional format. However, we are unaware of methods that perform representations in a Ł$_\infty$-based logical system which require less conditions or information than the provided by regional format or do not apply lattice representations.

5.2 A particular case: Truncated linear functions

Let us show the possibility of representing a rational McNaughton function in Ł$_\infty$-MODSAT and develop a polynomial algorithm for computing such representation in the particular case that function is a *truncated linear polynomial* with rational coefficients, defined in the following.

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

$$p(\mathbf{x}) = \frac{a_0}{b_0} + \frac{a_1}{b_1}x_1 + \cdots + \frac{a_n}{b_n}x_n, \tag{8}$$

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

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

$$p^\#(\mathbf{x}) = \min\Big(1, \max\big(0, p(\mathbf{x})\big)\Big). \tag{9}$$

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 (8), 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

$$p_P(\mathbf{x}) = \sum_{j \in P} \frac{\alpha_j}{\beta_j} x_j; \qquad\qquad p_N(\mathbf{x}) = \sum_{j \in N} \frac{\alpha_j}{\beta_j} x_j. \tag{10}$$

Let $Z_j^p, Z_{1/\beta_j} \in \mathbb{P}$; for a set of indexes $J \in \{P, N\}$, define:

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

And then, define:

$$\begin{aligned} &\bar{\varphi}_J = \tilde{\varphi}_J, && \bar{\Phi}_J = \tilde{\Phi}_J, && \text{if } 0 \notin J; \\ &\bar{\varphi}_J = \alpha_0 Z_{\frac{1}{\beta_0}} \oplus \tilde{\varphi}_J, && \bar{\Phi}_J = \tilde{\Phi}_J \cup \{\varphi_{\frac{1}{\beta_0}}\}, && \text{otherwise.} \end{aligned} \tag{11}$$

Lemma 8. *Functions p_P and p_N in (10) may, respectively, be represented by $\langle \bar{\varphi}_P, \bar{\Phi}_P \rangle$ and $\langle \bar{\varphi}_N, \bar{\Phi}_N \rangle$ in (11).*

Proof. See Preto and Finger (2020, Lemma 5.2). □

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

$$\bar{\varphi}_p = \beta[\neg(\bar{\varphi}_P \to \bar{\varphi}_N)], \qquad\qquad \bar{\Phi}_p = \bar{\Phi}_P \cup \bar{\Phi}_N. \tag{12}$$

Theorem 9. *Function $p^\#$ in (9) may be represented by $\langle \bar{\varphi}_p, \bar{\Phi}_p \rangle$ in (12).*

Proof. See Preto and Finger (2020, 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 they denote n-fold repetitions of a formula ψ. We deviate from this situation by using $\lfloor \log n \rfloor + 1$ new propositional variables $\xi_\psi^0, \xi_\psi^1, \ldots, \xi_\psi^{\lfloor \log n \rfloor}$ and replacing every occurrence of $n\psi$, where $n \in \mathbb{N} \setminus \{0, 1\}$, with the formula

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

$$\xi_{n\psi} =_{\text{def}} \bigoplus_{\substack{k=0 \\ n_k=1}}^{\lfloor \log n \rfloor} \xi_{\psi}^{k}, \tag{13}$$

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 $\bar{\Phi}_p$:

$$\begin{aligned} &\xi_{\psi}^{0} \leftrightarrow \psi; \\ &\xi_{\psi}^{k} \leftrightarrow \xi_{\psi}^{k-1} \oplus \xi_{\psi}^{k-1}, \text{ for } k = 1, \ldots, \lfloor \log n \rfloor. \end{aligned} \tag{14}$$

These formulas define the propositional variables ξ_{ψ}^{k} and we call $\Xi_{n\psi}$ the set that comprehends them. In this way, we avoid exponential blow up as shown in Theorem 10.

Theorem 10. *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 (12) by replacing any occurrence of $n\psi$ in $\bar{\varphi}_p$ and $\bar{\Phi}_p$ with $\xi_{n\psi}$ in (13) and by adding formulas in set $\Xi_{n\psi}$ in (14) to $\bar{\Phi}_p$. Then, $\langle \varphi_p, \Phi_p \rangle$ is also a representation for $p^{\#}$ in (9). Furthermore, $\langle \varphi_p, \Phi_p \rangle$ is a representation for $p^{\#}$ if it is defined by multiple suitable replacements of expressions $n_l\psi_l$, for $l = 1, \ldots, L$.*

Proof. See Preto and Finger (2020, Theorem 5.5). □

We set $\langle \varphi_p, \Phi_p \rangle$ from $\langle \bar{\varphi}_p, \bar{\Phi}_p \rangle$ in (12) 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

$$\varphi_p = \beta[\neg(\varphi_P \rightarrow \varphi_N)]; \qquad \Phi_p = \Phi_P \cup \Phi_N; \tag{15}$$

where φ_P, φ_N, Φ_P, and Φ_N are properly defined from their barred correspondents in (11). Table 2 shows how functions in Example 4 can be represented as in Theorem 10.

Algorithms 2 and 3 compute the representation of $n\psi$ in Ł$_\infty$-MODSAT in time $O(\log n)$ assuming that propositional variables are all represented with a constant size. Algorithm 2 returns $\mathbf{0}$ and ψ in the limit cases $n = 0$ and $n = 1$ (lines 1–5); when $n \in \mathbb{N} \setminus \{0, 1\}$, it returns formula $\xi_{n\psi}$ in (13) by building it in line 6 plus a $\lfloor \log n \rfloor + 1$ iteration loop (lines 7–13) where the n_k's in the binary representation of n are calculated by the routine in lines 8 and 9. Algorithm 3 returns $\emptyset$ in the limit cases $n = 0$ and $n = 1$ (lines 1–3); when $n \in \mathbb{N} \setminus \{0, 1\}$, it returns set $\Xi_{n\psi}$ that comprehends formulas (14) by building it in line 4 plus a $\lfloor \log n \rfloor$ iteration loop (lines 5–7). Algorithm 4 computes a representation of $p^{\#}$ in Ł$_\infty$-MODSAT. It returns $\langle \mathbf{0}, \emptyset \rangle$ in the limit case $a_0 = \cdots = a_n = 0$ (lines 1–3); otherwise it returns representation $\langle \varphi_p, \Phi_p \rangle$ given in (15). From lines 4 to 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. From lines 16 to 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 18–21) separately from the others (lines 22–25). In lines 27 and 28, it finally writes formula φ_p and completes set Φ_p.

Theorem 11. *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 4.*

Proof. See Preto and Finger (2020, Theorem 5.6). □

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 4.

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

Table 2. Representations as in (15) for functions $p_1^{\#}$, $p_2^{\#}$ and $p_3^{\#}$, where functions p_1, p_2, and p_3 are from Example 4

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

5.3 The general case

We can finally tackle the general case by means of a lattice representation. Let $f : [0, 1]^n \to [0, 1]$ be a rational McNaughton function in regional format with linear pieces:

$$p_i(\mathbf{x}) = \frac{a_{i0}}{b_{i0}} + \frac{a_{i1}}{b_{i1}}x_1 + \cdots + \frac{a_{in}}{b_{in}}x_n, \tag{16}$$

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

Algorithm 2 BINARY-F: computes formula $\xi_{n\psi}$ in (13) or **0** or ψ

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

if $n = 0$ **then**
  return **0**;
else if $n = 1$ **then**
  return ψ;
end if
$q := n$, $n_k := 0$, $\xi_{n\psi} := \mathbf{0}$;
for $k = 0, \ldots, \lfloor \log n \rfloor$ **do**
  $n_k :=$ remainder from division of q by 2;
  $q :=$ quotient from division of q by 2;
  if $n_k = 1$ **then**
    $\xi_{n\psi} := \xi_{\psi}^{k} \oplus \xi_{n\psi}$;
  end if
end for
return $\xi_{n\psi}$;

Algorithm 3 BINARY-S: computes set $\Xi_{n\psi}$ in (14) or $\emptyset$

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

if $n = 0$ or $n = 1$ **then**
  return $\emptyset$;
end if
$\Xi_{n\psi} := \left\{ \xi_{\psi}^{0} \leftrightarrow \psi \right\}$;
for $k = 1, \ldots, \lfloor \log n \rfloor$ **do**
  $\Xi_{n\psi} := \Xi_{n\psi} \cup \left\{ \xi_{\psi}^{k} \leftrightarrow \xi_{\psi}^{k-1} \oplus \xi_{\psi}^{k-1} \right\}$;
end for
return $\Xi_{n\psi}$;

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 time 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 10, for $i = 1, \ldots, m$. We define

$$\varphi = \bigvee_{i=1}^{m} \varphi_{\Omega_i}, \text{ with } \varphi_{\Omega_i} = \bigwedge_{k \in K_{\Omega_i}} \varphi_{p_k}; \qquad \Phi = \bigcup_{i=1}^{m} \Phi_{p_i}; \tag{17}$$

where $k \in K_{\Omega_i}$ iff p_k is above p_i over Ω_i.

Lemma 12. *Let f be a rational McNaughton function in regional format with linear pieces given by (16) and let φ_{Ω_j} be a formula and Φ a set as in (17). Then, $v(\varphi_{\Omega_j}) \leq f(v(X_1), \ldots, v(X_n))$, for $v \in \mathbf{Val}_{\Phi}$.*

Proof. Let $v \in \mathbf{Val}_{\Phi}$ and $\mathbf{x}_0 = \langle v(X_1), \ldots, v(X_n) \rangle$. In particular, $v \in \mathbf{Val}_{\Phi_{p_k}}$, for $k \in K_{\Omega_i}$ and, by Theorem 10, $v(\varphi_{\Omega_j}) = \min_{k \in K_{\Omega_j}} p_k^{\#}(\mathbf{x}_0)$. If $\mathbf{x}_0 \in \Omega_j$, then $v(\varphi_{\Omega_i}) \leq p_j^{\#}(\mathbf{x}_0) = p_j(\mathbf{x}_0) = f(\mathbf{x}_0)$. On the

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

Algorithm 4 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 $\langle \mathbf{0}, \emptyset \rangle$;
end if
$P := \emptyset, N := \emptyset$;
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 $\max\left\{\sum_{j \in P} \frac{a_j}{b_j}, \quad -\sum_{j \in N} \frac{a_j}{b_j}\right\}$;
for $j \in P \cup N$ **do**
  $\beta_j := \beta \cdot b_j$;
end for
$\varphi_P := \mathbf{0}, \varphi_N := \mathbf{0}, \Phi_p := \emptyset$;
for $J = P, N$ **do**
  if $0 \in J$ **then**
    $\varphi_J := \varphi_J \oplus \text{BINARY-F}\left(\alpha_0, Z_{1/\beta_0}\right)$;
    $\Phi_p := \Phi_p \cup \left\{Z_{1/\beta_0} \leftrightarrow \neg\text{BINARY-F}\left(\beta_0 - 1, Z_{1/\beta_0}\right)\right\} \cup \text{BINARY-S}\left(\alpha_0, Z_{1/\beta_0}\right) \cup \text{BINARY-S}\left(\beta_0 - 1, Z_{1/\beta_0}\right)$;
  end if
  for $j \in J \setminus \{0\}$ **do**
    $\varphi_J := \varphi_J \oplus \text{BINARY-F}\left(\alpha_j, Z_j^p\right)$;
    $\Phi_p := \Phi_p \cup \left\{Z_{1/\beta_j} \leftrightarrow \neg\text{BINARY-F}\left(\beta_j - 1, Z_{1/\beta_j}\right), \ \text{BINARY-F}\left(\beta_j, Z_j^p\right) \leftrightarrow X_j, Z_j^p \rightarrow Z_{1/\beta_j}\right\} \cup \text{BINARY-S}\left(\alpha_j, Z_j^p\right) \cup \text{BINARY-S}\left(\beta_j - 1, Z_{1/\beta_j}\right) \cup \text{BINARY-S}\left(\beta_j, Z_j^p\right)$;
  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 $\langle \varphi_p, \Phi_p \rangle$;

other hand, if $\mathbf{x}_0 \notin \Omega_j$, there is some i such that $\mathbf{x}_0 \in \Omega_i$. By the lattice property of regional format, there is k_0 such that p_i is above p_{k_0} over Ω_i and p_{k_0} is above p_j in Ω_j, then $k_0 \in K_{\Omega_j}$ and $v(\varphi_{\Omega_j}) \leq p^{\#}_{k_0}(\mathbf{x}_0) \leq p^{\#}_i(\mathbf{x}_0) = p_i(\mathbf{x}_0) = f(\mathbf{x}_0)$. □

Theorem 13. *Any rational McNaughton function may be represented by* $\langle \varphi, \Phi \rangle$ *in* (17).

Proof. First note that any rational McNaughton function may be put in regional format as showed in Section 5.1. 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 $v\left(Z_j^{p_i}\right) = \frac{x_j}{\beta_{ij}}$, for $i = 1, \ldots, m, j = 1, \ldots, n$, $v\left(Z_{1/\beta_{ij}}\right) = \frac{1}{\beta_{ij}}$, for $i = 1, \ldots, m, j = 0, \ldots, n$, $v\left(\xi^0_\psi\right) = v(\psi)$ and $v\left(\xi^k_\psi\right) = \min\left(1, v\left(\xi^{k-1}_\psi\right) + v\left(\xi^{k-1}_\psi\right)\right)$, for $k = 1, \ldots, \lfloor \log n \rfloor$, for any $n\psi$ that

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

Table 3. Representation as in (17) for function f from Example 4

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:	$\Phi_{p_1} \cup \Phi_{p_2} \cup \Phi_{p_3}$

Algorithm 5 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.

1: $\Phi := \emptyset$;
2: **for** $i = 1, \ldots, m$ **do**
3: $\quad \varphi_{p_i} :=$ REPRESENT-TL-F $\left(\frac{a_{i0}}{b_{i0}}, \ldots, \frac{a_{in}}{b_{in}}\right)$;
4: $\quad \varphi_{\Omega_i} := \varphi_{p_i}$;
5: **end for**
6: **for** $i = 1, \ldots, m$ **do**
7: $\quad$ **for** $k = 1, \ldots, i-1, i+1, \ldots, m$ **do**
8: $\quad\quad$ **if** ABOVE$(p_k, p_i) =$ **true then**
9: $\quad\quad\quad \varphi_{\Omega_i} = \varphi_{\Omega_i} \wedge \varphi_{p_k}$;
10: $\quad\quad$ **end if**
11: $\quad$ **end for**
12: $\quad \Phi := \Phi \cup$ REPRESENT-TL-S $\left(\frac{a_{i0}}{b_{i0}}, \ldots, \frac{a_{in}}{b_{in}}\right)$;
13: **end for**
14: $\varphi := \varphi_{\Omega_1} \vee \cdots \vee \varphi_{\Omega_m}$;
15: **return** $\langle \varphi, \Phi \rangle$;

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 10, $v(\varphi_{p_i}) = v'(\varphi_{p_i})$, 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 12. □

Table 3 shows how function f in Example 4 can be represented as in Theorem 13.

Algorithm 5 returns representation $\langle \varphi, \Phi \rangle$ for function f with linear pieces given in (16). From lines 1 to 13, the algorithm writes formulas φ_{Ω_i} and the set Φ: it first computes formulas φ_{p_i} (lines 2–5) by means of routine REPRESENT-TL-F and then it writes φ_{Ω_i} (lines 7–11) 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.

Theorem 14. *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 5.*

Proof. See Preto and Finger (2020, Theorem 6.3). □

5.4 Pre-regional format and a literature review

The algorithm presented for building representations in Ł$_\infty$-MODSAT comprehends two distinguished steps, the representation of the truncated version of linear pieces and the representation of the entire rational McNaughton function by means of a lattice representation. The second step

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

Table 4. Regions Ω_i for function f in Example 5

Ω_1	Ω_2	Ω_3	Ω_4	Ω_5
$1-x_1-x_2\geq 0$	$-1+x_1+x_2\geq 0$	$-x_1+x_2\geq 0$	$x_1-x_2\geq 0$	$1-x_1\geq 0$
$x_1\geq 0$	$\frac{1}{2}-x_1\geq 0$	$-\frac{1}{2}+x_1\geq 0$	$1-x_1\geq 0$	$\frac{1}{2}-x_2\geq 0$
$-\frac{1}{2}+x_2\geq 0$	$1-x_2\geq 0$	$1-x_2\geq 0$	$-\frac{1}{2}+x_2\geq 0$	$x_1\geq 0$
				$x_2\geq 0$

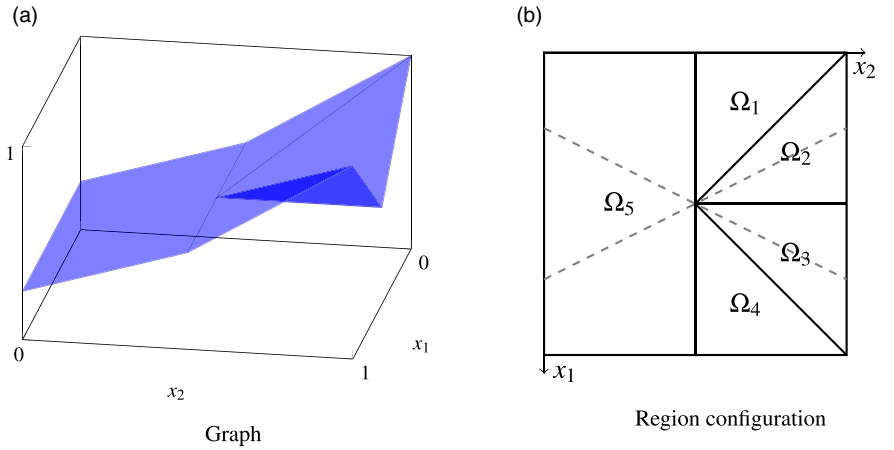


Figure 3. Graph and region configuration of function f_{CE} in Example 5.

is based on Lemma 12 and Theorem 13, where the encoding of the input function in regional format is required to comply with the lattice property. In the following, we discuss the necessity of such property for the correctness of the construction.

We say that a rational McNaughton function is in *pre-regional format* if it satisfies the first three items of the definition of regional format in Section 5.1, but it does not necessarily satisfy the lattice property; thus, functions in regional format are also in pre-regional format; however, the converse is not necessarily true. The following example shows that the encoding of a rational McNaughton function in pre-regional format is not enough to assure that an actual representation in Ł$_\infty$-MODSAT is built by the algorithm proposed in the previous section.

Example 5. Rational McNaughton function f_{CE} with graph in Figure 3a may be given in pre-regional format by the linear pieces $p_1(x_1,x_2)=p_4(x_1,x_2)=x_2$, $p_2(x_1,x_2)=1-x_1$, $p_3(x_1,x_2)=x_1$ and $p_5(x_1,x_2)=\frac{1}{4}+\frac{1}{2}x_2$. Regions Ω_i associated to each linear piece are depicted in Figure 3b and described in Table 4. The intersection of hyperplane given by p_5 with the hyperplanes given by p_2 and p_3 are depicted by the dotted lines in Figure 3b.

Note that such encoding of f_{CE} does not have the lattice property since there is no linear piece p_k such that p_3 is above p_k over Ω_3 and p_k is above p_5 in Ω_5. Let $\langle\varphi,\Phi\rangle$ be a pair as in (17), intending to represent f_{CE}, and let $\mathbf{x}_0=\langle 0.6,0.9\rangle\in\Omega_3^\circ$; we have $K_{\Omega_3}=\{1,3\}$ and $K_{\Omega_5}=\{5\}$ and, then, for $v\in\mathbf{Val}_\Phi$ such that $\mathbf{x}_0=\langle v(X_1),v(X_2)\rangle$, we have $f_{CE}(\mathbf{x}_0)=p_3(\mathbf{x}_0)=0.6<0.7=v(\varphi_{\Omega_5})\leq v(\varphi)$. Therefore, $\langle\varphi,\Phi\rangle$ cannot be a representation for function f_{CE} and the lattice property cannot be dropped from regional format in order to perform such representation.

Function f_{CE} may be put in regional format by taking as regions the polyhedra $P_\rho\in\mathscr{C}$ in (7); in this case, we have a representation with $|\mathscr{C}|=9$ regions. On the other hand, it may be put in regional format from the encoding above by only splitting region Ω_5 in two regions $\Omega_5'=\Omega_5\cap\{p_i\geq 0\}$ and $\Omega_5''=\Omega_5\cap\{p_i\leq 0\}$, for some $i\in\{2,3\}$, adding only one more region to the encoding.

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

The results in Tarela et al. (1990, Theorem 7), Tarela and Martínez (1999, Theorem 4.2) and, more recently, in Xu and Wang (2019, Theorem 1) propose a lattice representation of piecewise linear functions analogous to the one we derived in Lemma 12 and Theorem 13, where φ is a $(\bigvee \bigwedge)$-combination of formulas φ_{p_k}; they are presented in a more general context of piecewise linear functions over more general domains and codomains and do not refer to a specific formal language. However, those results do not require that the configuration of regions and linear pieces in the function description have the lattice property; thus, rational McNaughton functions only in pre-regional format would be enough for applying such results in our context. Unfortunately, despite being a less restrictive hypothesis, it is not actually suitable for our kind of representation, as Example 5 demonstrates. Nevertheless, this less restrictive approach is suitable for one-variable piecewise linear functions in pre-regional format due to the fact that functions in such encoding already have the lattice property.

Theorem 15. *One-variable rational McNaughton functions in pre-regional format have the lattice property; i.e., they are also in regional format.*

Proof. With no loss of generality, we may consider the regions of a one-variable rational McNaughton function in pre-regional format $f : [0, 1] \to [0, 1]$ to be nonempty closed intervals $[a, b] \subset [0, 1]$. Let $\Omega_i = [a_i, b_i]$ and $\Omega_j = [a_j, b_j]$ be regions such that $b_i \leq a_j$. If neither p_i is above p_j over Ω_i nor p_i is above p_j over Ω_j (then, $b_i < a_j$), let $P_\sigma, P_\varsigma \in \mathscr{C}$ be polyhedra as in (7) such that there are $\alpha, \beta \in [0, 1]$ in a way that $[\alpha, b_i] \subset P_\sigma$, $[a_j, \beta] \subset P_\varsigma$ and $(\alpha, b_i) \neq \emptyset \neq (a_j, \beta)$. For $\alpha' \in (\alpha, b_i)$ and $\beta' \in (a_j, \beta)$, let $\mathbf{X} = \langle \alpha', f(\alpha') \rangle$, $\mathbf{Y} = \langle \beta', f(\beta') \rangle$ and $[\mathbf{X}, \mathbf{Y}] = \{(1 - \lambda)\mathbf{A} + \lambda\mathbf{B} \mid \lambda \in [0, 1]\}$ be a line segment in $[0, 1]^2$. By our assumptions about p_i and p_j, p_i is strictly below $[\mathbf{X}, \mathbf{Y}]$ over (α', b_i) and p_j is strictly above $[\mathbf{X}, \mathbf{Y}]$ over (a_j, β'). Then, among all $c \in (b_i, a_j)$ such that $\langle c, f(c) \rangle \in [\mathbf{A}, \mathbf{B}]$, there is some d nearest to α' (which cannot be β'); let p_k be a linear piece such that $\langle d, f(d) \rangle = \langle d, p_k(d) \rangle \in [\mathbf{X}, \mathbf{Y}]$ and f coincides with p_k on some nonempty interval (d', d). For $x < d$, $p_k(x)$ is strictly below $[\mathbf{X}, \mathbf{Y}]$ and, for $x > d$, $p_k(x)$ is strictly above $[\mathbf{X}, \mathbf{Y}]$ and, then, p_i is above p_k over Ω_i and p_k is above p_j over Ω_j. Therefore, f has the lattice property, and it is given in regional format. The result is analogous for the case where $b_j \leq a_i$. □

6. Implementation and Results

We have developed a C++-implementation of Algorithms 4 and 5 for building representations of functions; it consists of two main modules. One module builds a representation for the truncated linear function $p^\#$ as in (9) from a given linear function as in (8). The other module encompasses the first one and builds a representation for a piecewise linear function f in closed regional format given by linear pieces as in (8) which are identical to f in given polyhedral regions as in (5). The routine for deciding whether linear piece p_k is above linear piece p_i over region Ω_i is the one in Algorithm 1 which was implemented using the C++ interface to the SoPlex linear programming solver (Gamrath et al., 2020).

We ran the implementation through experiments in order to measure its execution time and to give evidence for its correctness. The totality of a finite set of tests does not prove correctness; however, in large amounts, it may provide some evidence in favor of it.

In each experiment, the implementation was fed with a piecewise linear function f of n variables. Its execution time was measured and, with output $\langle \varphi, \Phi \rangle$, for random values $x_1, \ldots, x_n \in [0, 1]$, a valuation $v \in \mathbf{Val}_\Phi$ was computed such that $v(X_1) = x_1, \ldots, v(X_n) = x_n$. Finally, it was attested whether $v(\varphi) = f(x_1 \ \ldots, x_n)$ by separately evaluating φ and the original function f. Valuations v were computed using a $Ł_\infty$-solver based on the one by Ansótegui et al. (2012); it was written in the SMT-LIB language (Barrett et al., 2016) and ran in the Yices SMT solver (Dutertre, 2014).

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

Table 5. Number of tests by class of rational McNaughton functions

Class of functions	Tested functions	Evaluations per function	Evaluations
Truncated linear	5.000	100	500.000
Normalized linear	5.000	100	500.000
Simple-region piecewise linear	1.000	100	100.000
Cubic-region piecewise linear	990	100	99.000
Total	11.990	100	1.199.000

We ran four batteries of experiments, each one comprehending functions belonging to a class of rational McNaughton functions which were randomly generated according to a specification; in any case, each function was evaluated in 100 combinations of random values $x_1, \ldots, x_n \in [0, 1]$, which were uniformly chosen over the interval $[0, 1]$. Table 5 summarizes the experiments.

All the experiments in this section were run in a UNIX machine with two E5645 CPUs @ 2.40GHz with 12 processors. The source code for the implementation and the experiments are publicly available.[I]

6.1 Classes of rational McNaughton functions and experiments

Following, we describe the classes of functions we used in each battery of experiments and the specifications according to which random functions in these classes were generated. Before that, we state a result on continuous piecewise linear functions which we assume in the constructions in the latter classes.

Theorem 16. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuous piecewise linear function identical to $p_1 : \mathbb{R}^n \to \mathbb{R}$ and $p_2 : \mathbb{R}^n \to \mathbb{R}$ over $R_1 \subset \mathbb{R}^n$ and $R_2 \subset \mathbb{R}^n$, respectively. If p_1 and p_2 have rational coefficients and*

$$R_1 \cap R_2 = \left\{ \langle x_1, \ldots, x_n \rangle \in \mathbb{R}^n \;\middle|\; x_{j_0} = \xi,\ \alpha_j \leq x_j \leq \beta_j, \text{ for } j = 1, \ldots, j_0 - 1, j_0 + 1, \ldots, n \right\},$$

for $\xi, \alpha_j, \beta_j \in \mathbb{Q}$, with $\xi \neq 0$ and $\alpha_j < \beta_j$, for $j = 1, \ldots, j_0 - 1, j_0 + 1, \ldots, n$, then, there is $q \in \mathbb{Q}$, such that $p_1(\mathbf{x}) - p_2(\mathbf{x}) = q \cdot (x_{j_0} - \xi)$, for $\mathbf{x} \in \mathbb{R}^n$.

Proof. Let $p_i(\mathbf{x}) = \gamma_{i0} + \gamma_{i1}x_1 + \cdots + \gamma_{in}x_n$, for $i = 1, 2$ and $\mathbf{x} = \langle x_1, \ldots, x_n \rangle \in \mathbb{R}^n$. Since $p_1(\mathbf{x}_0) = p_2(\mathbf{x}_0)$, for any $\mathbf{x}_0 \in R_1 \cap R_2$, we must have that $\gamma_{1j} = \gamma_{2j}$, for $j = 2, \ldots, j_0 - 1, j_0 + 1, \ldots, n$, and $(\gamma_{10} - \gamma_{20}) + (\gamma_{1j_0} - \gamma_{2j_0})\xi = 0$. The result follows by letting $q = \frac{\gamma_{20} - \gamma_{10}}{\xi}$. □

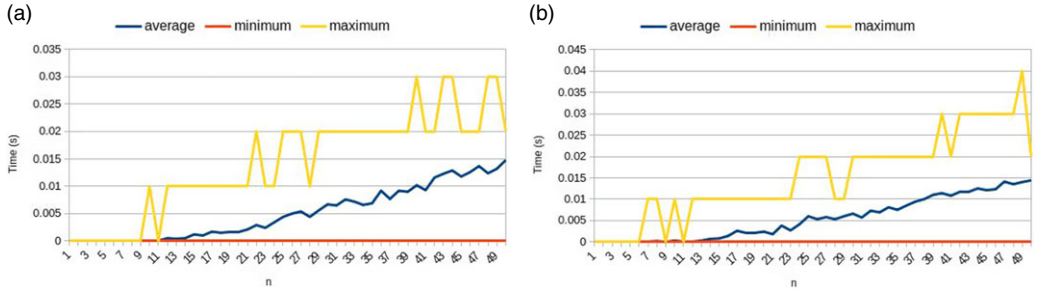
Truncated linear functions. A function $p^{\#} : [0, 1]^n \to [0, 1]$ in this class is a truncated linear function in (9) defined from a linear function p in (8). Function $p^{\#}$ has range in $[0, 1]$ and is continuous over $[0, 1]^n$.

In the experiments, for each dimension $n = 1, \ldots, 50$, one hundred functions $p^{\#}$ were generated from functions p for which, for each coefficient $\frac{a_j}{b_j}$, a_j was randomly chosen among integers from -100 to 100 and b_j was randomly chosen among integers from 1 to 100. The execution time for building the representations in $Ł_\infty$-MODSAT was up to 0.03 second. In Figure 4a, we see the results of the representation builder running on truncated linear functions.

Normalized linear functions. A function $p' : [0, 1]^n \to [0, 1]$ in this class is defined from a linear function p in (8) by the following normalization process performed over $D = [0, 1]^n$ by

$$p'(\mathbf{x}) = \frac{p(\mathbf{x}) + \frac{A}{b_0}}{B}, \tag{18}$$

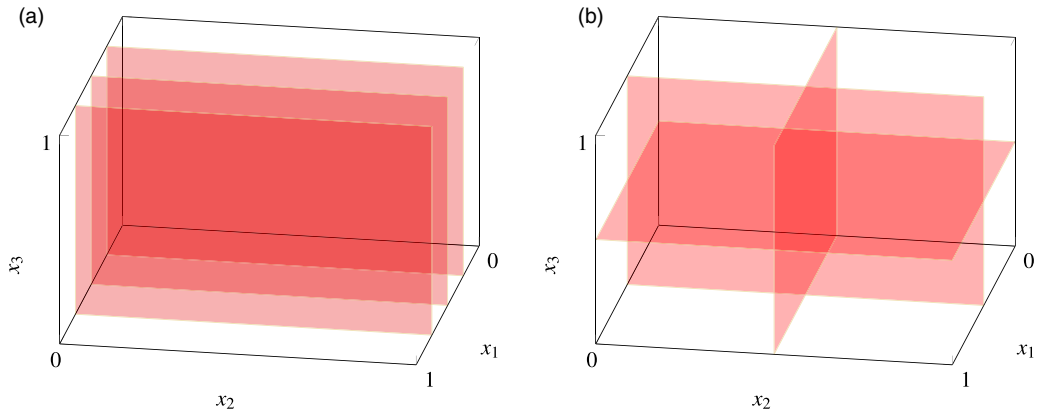
https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press



Running on truncated linear functions

Running on normalized linear functions

Figure 4. Representation builder performance, randomly gen. instances: $n = 1$ to $n = 50$.



Simple-region config.

Cubic-region config.

Figure 5. Simple-region and cubic-region configurations in dimension $n = 3$.

for $\mathbf{x} \in [0,1]^n$, where A is the least positive integer such that $\frac{A}{b_0} \geq |\min_{\mathbf{x}\in D} p(\mathbf{x})|$, if $\min_{\mathbf{x}\in D} p(\mathbf{x}) < 0$, and $A = 0$, otherwise; and B is the least integer greater than or equal to $\max_{\mathbf{x}\in D} p(\mathbf{x}) + \frac{A}{b_0}$, if $\max_{\mathbf{x}\in D} p(\mathbf{x}) + \frac{A}{b_0} > 1$, and $B = 1$, otherwise. Function p' has range in $[0,1]$ and is continuous over $[0,1]^n$.

In the experiments, for each dimension $n = 1, \ldots, 50$, one hundred functions p' were generated from functions p for which, for each coefficient $\frac{a_j}{b_j}$, a_j was randomly chosen among integers from -100 to 100 and b_j was randomly chosen among integers from 1 to 100. The execution time for building the representations in Ł$_\infty$-MODSAT was up to 0.04 second. In Figure 4b, we see the results of the representation builder running on normalized linear functions.

Simple-region piecewise linear functions. A function $f : [0,1]^n \to [0,1]$ in this class is defined to be identical to linear pieces p_i over (simple-)regions

$$\Omega_i = \left\{ \mathbf{x} = \langle x_1, \ldots, x_n \rangle \in [0,1]^n \;\middle|\; \frac{i-1}{r} \leq x_1 \leq \frac{i}{r},\ 0 \leq x_j \leq 1,\ \text{for } j = 2, \ldots, n \right\},$$

for $i = 1, \ldots, r$. Figure 5a depicts a simple-region configuration with four regions for $n = 3$ and $r = 4$.

Linear piece p_1 is defined by p' from a linear function p in (8) by the normalization process in (18) performed over $D = \Omega_1$.

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

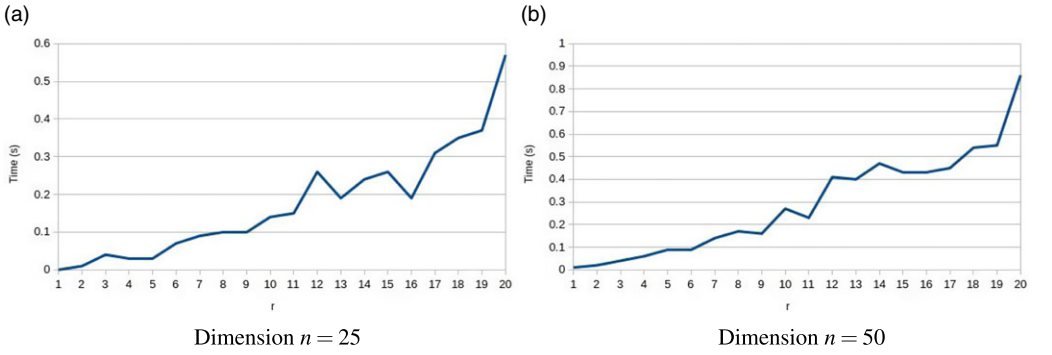


Figure 6. Representation builder performance running on simple-region piecewise linear functions, randomly gen. instances: $r = 1$ to $r = 20$.

The other linear pieces p_i, for $i = 2, \ldots, r$, are defined by

$$p_i(\mathbf{x}) = p_{i-1}(\mathbf{x}) + q_i \cdot \left(x_1 - \frac{i-1}{r}\right),$$

with $q_i \in [-m_i \cdot r, (1 - M_i) \cdot r]$, for

$$m_i = \min_{\substack{\mathbf{x}\in\Omega_i \\ \text{s.t. } x_1 = \frac{i}{r}}} p_{i-1}(\mathbf{x}) \qquad \text{and} \qquad M_i = \max_{\substack{\mathbf{x}\in\Omega_i \\ \text{s.t. } x_1 = \frac{i}{r}}} p_{i-1}(\mathbf{x}).$$

These linear pieces and, therefore, function f have range in $[0, 1]$; also, function f is continuous over $[0, 1]^n$. Theorem 17 below states that such encoding of function f has the lattice property.

In the experiments, for each dimension $n = 1, \ldots, 50$ and each number of regions $r = 1, \ldots, 20$, one function f was generated with linear piece p_1 defined from a function p for which, for each coefficient $\frac{a_j}{b_j}$, a_j was randomly chosen among integers from -100 to 100 and b_j was randomly chosen among integers from 1 to 100; and with linear pieces p_i defined from linear pieces p_{i-1} and values q_i uniformly chosen over the intervals $[-m_i \cdot r, (1 - M_i) \cdot r]$, for $i = 2, \ldots, r$. The execution time for building the representations in Ł$_\infty$-MODSAT was up to 1 second. In Figure 6, we see the results of the representation builder running on simple-region piecewise linear functions with dimensions $n = 25$ and $n = 50$.

Cubic-region piecewise linear functions. A function $f' : [0, 1]^n \to [0, 1]$ in this class is defined from a function $f : [0, 1]^n \to \mathbb{R}$ by the following normalization process performed by

$$f'(\mathbf{x}) = \frac{f(\mathbf{x}) + \gamma}{\Gamma},$$

for $\mathbf{x} \in [0, 1]^n$, where $\gamma = |\min_{\mathbf{x}\in[0,1]^n} f(\mathbf{x})|$, if $\min_{\mathbf{x}\in[0,1]^n} f(\mathbf{x}) < 0$, and $\gamma = 0$, otherwise; and Γ is the least integer greater than or equal to $\max_{\mathbf{x}\in[0,1]^n} f(\mathbf{x}) + \gamma$, if $\max_{\mathbf{x}\in[0,1]^n} f(\mathbf{x}) + \gamma > 1$, and $\Gamma = 1$, otherwise. Function f' has range in $[0, 1]$.

Function $f : [0, 1]^n \to \mathbb{R}$ is defined to be identical to linear pieces $p_{\langle i_1,\ldots,i_n\rangle}$ over (cubic-)regions

$$\Omega_{\langle i_1,\ldots,i_n\rangle} = \left\{\mathbf{x} = \langle x_1, \ldots, x_n\rangle \in [0, 1]^n \;\middle|\; \frac{i_j - 1}{r} \le x_j \le \frac{i_j}{r}, \text{ for } j = 1, \ldots, n\right\},$$

for $i_j = 1, \ldots, r$, for $j = 1, \ldots, n$. Figure 5b depicts a cubic-region configuration with eight regions for $n = 3$ and $r = 2$.

Linear piece $p_{\langle 1,\ldots,1\rangle}$ is defined by p' from a linear function p in (8) by the normalization process in (18) performed over $D = \Omega_{\langle 1,\ldots,1\rangle}$.

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

The linear pieces $p_{\langle i_1,\ldots,i_n\rangle}$, for which $i_1 = \cdots = i_{j-1} = i_{j+1} = \cdots = i_n = 1$ and $i_j \neq 1$, are defined by

$$p_{\langle i_1,\ldots,i_n\rangle}(\mathbf{x}) = p_{\langle i_1,\ldots,i_{j-1},i_j-1,i_{j+1},\ldots,i_n\rangle}(\mathbf{x}) + q_j^{i_j} \cdot \left(x_j - \frac{i_j - 1}{r}\right), \tag{19}$$

with $q_j^{i_j} \in [-m_{\langle i_1,\ldots,i_n\rangle} \cdot r, (1 - M_{\langle i_1,\ldots,i_n\rangle}) \cdot r]$, for

$$m_{\langle i_1,\ldots,i_n\rangle} = \min_{\mathbf{x}\in\Omega} p_{\langle i_1,\ldots,i_{j-1},i_j-1,i_{j+1},\ldots,i_n\rangle}(\mathbf{x}) \quad \text{and} \quad M_{\langle i_1,\ldots,i_n\rangle} = \max_{\mathbf{x}\in\Omega} p_{\langle i_1,\ldots,i_{j-1},i_j-1,i_{j+1},\ldots,i_n\rangle}(\mathbf{x}),$$

where

$$\Omega = \left\{\mathbf{x} = \langle x_1,\ldots,x_n\rangle \in \Omega_{\langle i_1,\ldots,i_n\rangle} \;\middle|\; x_j = \frac{i_j}{r}\right\}.$$

These linear pieces already have range in $[0, 1]$ and function f is continuous over $\Omega_{\langle i_1,\ldots,i_n\rangle} \cap \Omega_{\langle i_1,\ldots,i_{j-1},i_j-1,i_{j+1},\ldots,i_n\rangle}$.

The other linear pieces $p_{\langle i_1,\ldots,i_n\rangle}$, for which $i_1 = \cdots = i_{j-1} = i_{j+1} = \cdots = i_{k-1} = 1$ and $i_j \neq 1 \neq i_k$, are also defined by (19) with the same $q_j^{i_j}$. These linear pieces are not guaranteed to have range in $[0, 1]$; however, function f is continuous over $\Omega_{\langle i_1,\ldots,i_n\rangle} \cap \Omega_{\langle i_1,\ldots,i_{j-1},i_j-1,i_{j+1},\ldots,i_n\rangle}$. It is also continuous over $\Omega_{\langle i_1,\ldots,i_n\rangle} \cap \Omega_{\langle i_1,\ldots,i_{l-1},i_l-1,i_{l+1},\ldots,i_n\rangle}$, for $l \geq k$; indeed, there is a value q such that

$$p_{\langle i_1,\ldots,i_n\rangle}(\mathbf{x}) = p_{\langle i_1,\ldots,i_{j-1},i_j-1,i_{j+1},\ldots,i_{l-1},i_l-1,i_{l+1},\ldots,i_n\rangle}(\mathbf{x}) + q \cdot \left(x_l - \frac{i_l - 1}{r}\right) + q_j^{i_j} \cdot \left(x_j - \frac{i_j - 1}{r}\right)$$

and we are able to write

$$p_{\langle i_1,\ldots,i_n\rangle}(\mathbf{x}) = p_{\langle i_1,\ldots,i_{l-1},i_l-1,i_{l+1},\ldots,i_n\rangle}(\mathbf{x}) + q \cdot \left(x_l - \frac{i_l - 1}{r}\right).$$

Thus, functions f and f' are continuous over $[0, 1]^n$. Theorem 17 below states that such encoding of function f' has the lattice property.

In the experiments, for each dimension $n = 1,\ldots,9$ and each regional parameter $r = 1,\ldots,7-(n-1)$, if $n \leq 5$, and $r = 1, 2$, otherwise, thirty functions f' were generated from functions f with linear piece $p_{\langle 1,\ldots,1\rangle}$ defined from a function p for which, for each coefficient $\frac{a_j}{b_j}$, a_j was randomly chosen among integers from -30 to 30 and b_j was randomly chosen among integers from 1 to 30; and with linear pieces $p_{\langle i_1,\ldots,i_n\rangle}$, for which $i_1 = \cdots = i_{j-1} = i_{j+1} = \cdots = i_n = 1$ and only $i_j \neq 1$, defined from linear pieces $p_{\langle i_1,\ldots,i_{j-1},i_j-1,i_{j+1},\ldots,i_n\rangle}$ and values $q_{\langle i_1,\ldots,i_n\rangle}$ uniformly chosen over the intervals $[-m_{\langle i_1,\ldots,i_n\rangle} \cdot r, (1 - M_{\langle i_1,\ldots,i_n\rangle}) \cdot r]$. In Table 6, we see the results of the representation builder running on cubic-region piecewise linear functions.

Theorem 17. *Simple-region and cubic-region piecewise linear functions in the presented encoding have the lattice property.*

Proof. Let Ω_i and Ω_j be simple regions of a simple-region piecewise linear function f. Fixing $x_2 = \xi_2 \in [0, 1],\ldots, x_n = \xi_n \in [0, 1]$, we define the restriction of f to $g : [0, 1] \to [0, 1]$ given by $g(x_1) = f(x_1, \xi_2,\ldots,\xi_n)$, which is a piecewise liner function with the lattice property by Theorem 15. Since, by Theorem 16, linear pieces of simple-region piecewise linear functions intercept each other over domain points in some set $\{\mathbf{x} \in [0, 1]^n \mid x_1 = K \in \mathbb{R}\}$, f also has the lattice property. Now, let $\Omega_{\langle i_1,\ldots,i_n\rangle}$ and $\Omega_{\langle I_1,\ldots,I_n\rangle}$ be cubic-regions of a cubic-region piecewise linear function f'; since the normalization process from f to f' does not interfere with the lattice property, we only need to show that f has the lattice property. Analogous to the previous argument for simple-regions, for $j = 1,\ldots,n$, there is k_j, for which $\min\{i_j, I_j\} \leq k_j \leq \max\{i_j, I_j\}$, such that $p_{\langle i_1,\ldots,i_n\rangle}(\mathbf{x}) \geq$

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

Table 6. Representation builder performance running on cubic-region piecewise linear functions

n	*r*	avgTime(s)	minTime(s)	maxTime(s)	*n*	*r*	avgTime(s)	minTime(s)	maxTime(s)
1	1	0	0	0	3	5	3.2877	2.64	6.4
1	2	0	0	0	4	1	0	0	0
1	3	0	0	0	4	2	0.0607	0.04	0.08
1	4	0	0	0	4	3	1.4173	1.08	1.92
1	5	0.001	0	0.01	4	4	15.5163	13.17	25.38
1	6	0.0013	0	0.01	5	1	0	0	0
1	7	0.0043	0	0.01	5	2	0.2423	0.17	0.4
2	1	0	0	0	5	3	15.284	11.55	27.38
2	2	0	0	0	6	1	0	0	0
2	3	0.0103	0	0.02	6	2	1.004	0.74	2.09
2	4	0.0463	0.03	0.08	7	1	0	0	0
2	5	0.1213	0.08	0.15	7	2	4.47	3.4	7.93
2	6	0.2427	0.19	0.32	8	1	0	0	0
3	1	0	0	0	8	2	19.4273	15.01	46.63
3	2	0.01	0	0.02	9	1	0	0	0
3	3	0.1473	0.11	0.19	9	2	85.9193	67.39	169.99
3	4	0.8707	0.67	1.05					

$p_{\langle i_1,\ldots,i_{j-1},k_j,i_{j+1},\ldots,i_n\rangle}(\mathbf{x})$, for $\mathbf{x} \in \Omega_{\langle i_1,\ldots,i_n\rangle}$, and $p_{\langle i_1,\ldots,i_{j-1},k_j,i_{j+1},\ldots,i_n\rangle}(\mathbf{x}) \geq p_{\langle i_1,\ldots,i_{j-1},I_j,i_{j+1},\ldots,i_n\rangle}(\mathbf{x})$, for $\mathbf{x} \in \Omega_{\langle i_1,\ldots,i_{j-1},I_j,i_{j+1},\ldots,i_n\rangle}$. Then, from the general formula for linear pieces

$$p_{\langle i_1\ldots,i_n\rangle}(\mathbf{x}) = p_{\langle 1,\ldots,1\rangle}(\mathbf{x}) + \sum_{j=1}^{n}\sum_{\iota=2}^{i_j} q_j^{\iota}\left(x_j - \frac{\iota-1}{r}\right),$$

it follows that $p_{\langle i_1,\ldots,i_n\rangle}(\mathbf{x}) \geq p_{\langle k_1,\ldots,k_n\rangle}(\mathbf{x})$, for $\mathbf{x} \in \Omega_{\langle i_1,\ldots,i_n\rangle}$, and $p_{\langle k_1,\ldots,k_n\rangle}(\mathbf{x}) \geq p_{\langle I_1,\ldots,I_n\rangle}(\mathbf{x})$, for $\mathbf{x} \in \Omega_{\langle I_1,\ldots,I_n\rangle}$. Therefore, f has the lattice property. □

7. Conclusions and Future Work

We investigated implicit representations of rational McNaughton functions by logical formulas in the Łukasiewicz Infinitely-valued Logic by means of semantics modulo satisfiability. We carried out a comparative investigation on different approaches to define such a representation concept, the formula-based approach, and the function-based approach, which was originally introduced by Finger and Preto (2020).

Rational McNaughton functions were constructively shown to be representable in $Ł_\infty$-MODSAT, which yielded a polynomial algorithm for building the representations. An implementation of the Algorithm 5 together with results of experimental tests were presented and, in order to set up the tests, we established classes of rational McNaughton functions from where random such functions may easily be chosen. In comparison with the existing literature, we were able to conclude that our approach has the advantage of efficiently building a representations in the Łukasiewicz Infinitely-valued Logicframework, whose associated problems remain within the NP boundary and about which there is considerable literature.

For the future, both the algorithm for building representations in $Ł_\infty$-MODSAT and its implementation might be improved in order to achieve efficiency gains. Also, we hope to couple this algorithm with algorithms that approximate (normalized) continuous functions by rational McNaughton functions.

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

Moreover, we might apply these approximations to the study of real systems such as neural networks through automated reasoning techniques since, depending on its class of activation functions, a neural network may be seen either as a piecewise linear function or as a continuous function that can be approximated by one (Leshno et al., 1993). Also, Amato et al. (2002) pointed that the representation of neural networks in a logical system may be useful in their interpretation, which is still a challenging task for research; thus, approximate representations in $Ł_\infty$-MODSAT might play a role in such endeavor.

Conflicts of interest. The authors declare none.

Note

1 http://github.com/spreto/pwl2limodsat.

References

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

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

Aguzzoli, S. and Mundici, D. (2003). *Weierstrass Approximation Theorem and Łukasiewicz Formulas with one Quantified Variable*, Physica-Verlag HD, Heidelberg, 315–335.

Amato, P., Di Nola, A. and Gerla, B. (2002). 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, 506–510.

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

Ansótegui, C., Bofill, M., Manyà, F. and Villaret, M. (2012). Building automated theorem provers for infinitely-valued logics with satisfiability modulo theory solvers. In: *2012 IEEE 42nd International Symposium on Multiple-Valued Logic*, 25–30.

Barrett, C., Fontaine, P. and Tinelli, C. (2016). The satisfiability modulo theories library (SMT-LIB). www.SMT-LIB.org.

Bertsimas, D. and Tsitsiklis, J. N. (1997). *Introduction to Linear Optimization*, Athena Scientific, Belmont, MA.

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

Cignoli, R., D'Ottaviano, I. and Mundici, D. (2000). *Algebraic Foundations of Many-Valued Reasoning*, Trends in Logic, Springer, Netherlands.

Di Nola, A. and Leuștean, I. (2011). Riesz MV-algebras and their logic. In: *Proceedings of the 7th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT-11)*, Atlantis Press, 140–145.

Di Nola, A. and Leuștean, I. (2014). Łukasiewicz logic and Riesz spaces. *Soft Computing* **18** 2349–2363.

Dutertre, B. (2014). Yices 2.2. In: Biere, A. and Bloem, R. (eds.) *Computer-Aided Verification (CAV'2014)*, Lecture Notes in Computer Science, vol. 8559, Springer, 737–744.

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

Finger, M. and Preto, S. (2018). Probably half true: Probabilistic satisfiability over Łukasiewicz infinitely-valued logic. In: Galmiche, D., Schulz, S. and Sebastiani, R. (eds.) *Automated Reasoning*, Cham, Springer International Publishing, 194–210.

Finger, M. and Preto, S. (2020). Probably partially true: Satisfiability for Łukasiewicz infinitely-valued probabilistic logic and related topics. *Journal of Automated Reasoning* **64** (7) 1269–1286.

Gamrath, G., Anderson, D., Bestuzheva, K., Chen, W.-K., Eifler, L., Gasse, M., Gemander, P., Gleixner, A., Gottwald, L., Halbig, K., Hendel, G., Hojny, C., Koch, T., Bodic, P. L., Maher, S. J., Matter, F., Miltenberger, M., Mühmer, E., Müller, B., Pfetsch, M., Schlösser, F., Serrano, F., Shinano, Y., Tawfik, C., Vigerske, S., Wegscheider, F., Weninger, D. and Witzig, J. (2020). The SCIP optimization suite 7.0. Technical report, Optimization Online.

Gerla, B. (2001). Rational Łukasiewicz logic and DMV-algebras. *Neural Network World* **11** (6) 579–594.

Leshno, M., Lin, V. Y., Pinkus, A. and Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks* **6** (6) 861–867.

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

Mundici, D. (1987). Satisfiability in many-valued sentential logic is NP-complete. *Theoretical Computer Science* **52** (1–2) 145–153.

Mundici, D. (1994). A constructive proof of McNaughton's theorem in infinite-valued logic. *Journal of Symbolic Logic* **59** (2) 596–602.

https://doi.org/10.1017/S096012952200010X Published online by Cambridge University Press

Preto, S. and Finger, M. (2020). An efficient algorithm for representing piecewise linear functions into logic. *Electronic Notes in Theoretical Computer Science* **351** 167–186. Proceedings of LSFA 2020, the 15th International Workshop on Logical and Semantic Frameworks, with Applications (LSFA 2020).

Tarela, J., Alonso, E. and Martínez, M. (1990). A representation method for PWL functions oriented to parallel processing. *Mathematical and Computer Modelling* **13** (10) 75–83.

Tarela, J. and Martínez, M. (1999). Region configurations for realizability of lattice piecewise-linear models. *Mathematical and Computer Modelling* **30** (11–12) 17–27.

Xu, J. and Wang, S. (2019). Lattice piecewise affine representations on convex projection regions. In: *2019 IEEE 58th Conference on Decision and Control (CDC)*, 7240–7245.

Cite this article: Preto S and Finger M (2022). Efficient representation of piecewise linear functions into Łukasiewicz logic modulo satisfiability. *Mathematical Structures in Computer Science* **32**, 1119–1144. https://doi.org/10.1017/S096012952200010X