Semantics modulo satisfiability with applications: function representation, probabilities and game theory

Sandro Márcio da Silva Preto

Thesis submitted
to the
Institute of Mathematics and Statistics
of the
University of São Paulo
in fulfillment of the requirements
for the degree
of
Doctor of Science

Program: Computer Science

Advisor: Prof. Dr. Marcelo Finger

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

São Paulo, June 2021

Semantics modulo satisfiability with applications: function representation, probabilities and game theory

This version of the thesis contains the corrections and changes suggested by the Examination Committee during the work's original version defence, which occurred on the 4th of June, 2021.

A copy of the original version is available at the Institute of Mathematics and Statistics of the University of São Paulo.

Examination Committee:

- Prof. Dr. Marcelo Finger (advisor) – IME-USP
- Prof. Dr. Fábio Gagliardi Cozman – Poli-USP
- Prof. Dr. Marcelo Esteban Coniglio – Unicamp
- Prof. Dr. João Marcos de Almeida – UFRN
- Prof. Dr. Eduardo Leopoldo Fermé – U. Madeira

To the memory of my father

Agradecimentos*

"Oi, pai, tudo bem?" "Tudo, Sandro, e por aí, como vão as coisas?" "Tudo bem também. Eu tô ligando pra dizer que saiu a minha bolsa!" "Nossa, mas essa é a melhor notícia do ano! Parabéns!" E foi mais ou menos assim que comemorei a boa notícia pelo telefone com o meu pai alguns dias depois de me mudar para São Paulo para começar meu doutorado. Infelizmente, não podemos comemorar juntos a conclusão desta jornada, mas deixo registrada esta gostosa lembrança da nossa relação, de que tanto tenho saudades, como uma pequena homenagem.

À minha mãe e ao meu pai, Rúbia e Marcio, sou profundamente grato. Toda conquista minha, inclusive esta tese, devo a eles por fazerem de mim quem sou, por seu entusiasmo, sua torcida, sua luta, seus sacrifícios e, principalmente, seu amor.

Agradeço também à minha irmã e à sua família. Núbia e Rafael, que sempre me proporcionam muito agradáveis momentos em minhas visitas à Guaranésia, e meus sobrinhos Afonso, a melhor e mais divertida companhia, e Otávio, ainda um bebê e que só é motivo de alegrias.

À minha namorada Fernanda, uma das melhores surpresas que a vida me proporcionou neste período, agradeço por todo carinho que me dedica, hoje essencial ao meu dia-a-dia, e também pelo apoio e pela compreensão, mesmo nos momentos um tanto angustiantes que compõem o doutorado.

Ao professor Marcelo, além da impecável dedicação à minha orientação, agradeço por ensinar que o bom trabalho pode ser feito de forma prazerosa e divertida e, mais que isso, por ser o necessário tipo de professor com sensibilidade e verdadeira preocupação com seus alunos. Deixo aqui também o meu apreço aos meus professores e aos servidores do IME-USP, cuja dedicação exemplifica as causas da excelência da universidade brasileira.

Por fim, agradeço a todos os meus amigos, antigos e novos, que de alguma forma fizeram parte deste trabalho. Pelas visitas em São Paulo, pelas caronas à Guaranésia, pelas trocas de mensagens no WhatsApp, pelas video-chamadas na pandemia, pelos bons momentos no laboratório, o LIAMF, ou nas mesas dos botecos, por me ouvirem, por me ensinarem, por sempre ampliarem meus horizontes, obrigado e *gracias*.

*Acknowledgments. Although it is quite reasonable to write a doctoral thesis in the lingua franca of science, I address those to whom I am grateful in our mother tongue (or, in some cases, even in their mother tongue).

Resumo

Sandro Márcio da Silva Preto. **Semântica módulo satisfatibilidade com aplicações: representação de funções, probabilidades e teoria dos jogos**. Tese (Doutorado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2021.

No contexto das lógicas proposicionais, aplicamos semânticas módulo satisfatibilidade — uma semântica restrita que contempla somente valorações que satisfazem algum conjunto específico de fórmulas — com o objetivo de resolver de forma eficiente algumas tarefas computacionais. Três destas possíveis aplicações são desenvolvidas.

Começamos estudando a possibilidade de representar implicitamente funções racionais de McNaughton na Lógica Infinito-valorada de Łukasiewicz por meio de semânticas módulo satisfatibilidade. Investigamos teoricamente algumas abordagens para este conceito de representação, chamado representação módulo satisfatibilidade, e descrevemos um algoritmo polinomial que constrói representações no sistema recém-introduzido. Uma implementação do algoritmo, resultados de testes e formas de gerar aleatoriamente funções racionais de McNaughton para os testes são apresentados. Mais que isso, propomos uma aplicação destas representações à verificação formal de propriedades de redes neurais através do ferramental de inferência da Lógica Infinito-valorada de Łukasiewicz.

Então, passamos a investigar a satisfatibilidade da atribuição conjunta de probabilidades a fórmulas da Lógica Infinito-valorada de Łukasiewicz, um problema sabidamente NP-completo. Fornecemos um algoritmo exato de decisão derivado da combinação de métodos de álgebra linear com semânticas módulo satisfatibilidade. Fornecemos também uma implementação para tal algoritmo para o qual o fenômeno da transição de fase é empiricamente detectado.

Por último, estudamos a situação em teoria dos jogos dos chamados jogos observáveis, que são jogos que sabidamente alcançam um equilíbrio de Nash, entretanto, um observador externo não conhece qual o exato perfil de ações que ocorre em uma instância específica; então, tal observador atribui probabilidades subjetivas às ações do jogadores. Estudamos o problema de decisão de determinar se um conjunto dessas restrições probabilísticas é coerente reduzindo-o aos problemas de satisfatibilidade de atribuições probabilísticas a fórmulas lógicas tanto na Lógica Proposicional Clássica quanto na Lógica Infinito-valorada de Łukasiewicz dependendo se somente equilíbrios puros são permitidos ou, também, equilíbrios mistos. Tais reduções dependem das propriedades das semânticas módulo satisfatibilidade. Oferecemos discussões sobre complexidade e algoritmos para o problema de coerência e, também, para o problema de computar restrições probabilísticas maximal e minimal sobre ações que preservem a coerência.

Palavras-chave: Semânticas de valoração, lógicas proposicionais, lógica infinito-valorada de Łukasiewicz, funções racionais de McNaughton, funções lineares por partes, representação de

funções, métodos formais, redes neurais, probabilidades não clássicas, satisfatibilidade probabilística, equilíbrio de Nash, jogos com incerteza, restrições probabilísticas, coerência de restrições.

Abstract

Sandro Márcio da Silva Preto. **Semantics modulo satisfiability with applications: function representation, probabilities and game theory**. Thesis (Doctorate). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2021.

In the context of propositional logics, we apply semantics modulo satisfiability — a restricted semantics which comprehends only valuations that satisfy some specific set of formulas — with the aim to efficiently solve some computational tasks. Three possible such applications are developed.

We begin by studying the possibility of implicitly representing rational McNaughton functions in Łukasiewicz Infinitely-valued Logic through semantics modulo satisfiability. We theoretically investigate some approaches to such representation concept, called representation modulo satisfiability, and describe a polynomial algorithm that builds representations in the newly introduced system. An implementation of the algorithm, test results and ways to randomly generate rational McNaughton functions for testing are presented. Moreover, we propose an application of such representations to the formal verification of properties of neural networks by means of the reasoning framework of Łukasiewicz Infinitely-valued Logic.

Then, we move to the investigation of the satisfiability of joint probabilistic assignments to formulas of Łukasiewicz Infinitely-valued Logic, which is known to be an NP-complete problem. We provide an exact decision algorithm derived from the combination of linear algebraic methods with semantics modulo satisfiability. Also, we provide an implementation for such algorithm for which the phenomenon of phase transition is empirically detected.

Lastly, we study the game theory situation of observable games, which are games that are known to reach a Nash equilibrium, however, an external observer does not know what is the exact profile of actions that occur in a specific instance; thus, such observer assigns subjective probabilities to players' actions. We study the decision problem of determining if a set of these probabilistic constraints is coherent by reducing it to the problems of satisfiability of probabilistic assignments to logical formulas both in Classical Propositional Logic and Łukasiewicz Infinitely-valued Logic depending on whether only pure equilibria or also mixed equilibria are allowed. Such reductions rely upon the properties of semantics modulo satisfiability. We provide complexity and algorithmic discussion for the coherence problem and, also, for the problem of computing maximal and minimal probabilistic constraints on actions that preserves coherence.

Keywords: Valuation semantics, propositional logics, Łukasiewicz infinitely-valued logic, rational McNaughton functions, piecewise linear functions, function representation, formal methods, neural networks, non-classical probabilities, probabilistic satisfiability, Nash equilibrium, uncertain games, probabilistic constraints, coherence of constraints.

Contents

List of Abbreviations

SAT	Satisfiability Problem, a computational problem
CPL	Classical Propositional Logic, a logical system
CNF	Conjunctive Normal Form
PSAT	Probabilistic Satisfiability Problem, a computational problem
P	Polynomial Class, a computational complexity class
NP	Nondeterministic Polynomial Class, a computational complexity class
ŁIP	Łukasiewicz Infinitely-valued Probabilistic
ŁIPSAT	Łukasiewicz Infinitely-valued Probabilistic Satisfiability Problem, a computational problem
PCE	Probabilistic Constraints on Equilibria
iff	"if, and only if"

List of Symbols

$\mathbf{L}$	A logical system
$\mathcal{L}$	A logical language
$\models$	A logical consequence relation
$\mathcal{V}$	A set of truth values
$\mathcal{D}$	A set of designated truth values
$\mathbf{Val}$	A (valuation) semantics
$\mathbf{Val}_\Phi$	A (valuation) semantics modulo satisfiability of the set Φ
$\mathrm{Th}(\Gamma)$	A logical theory with axioms in set Γ
$\mathbb{P}$	The set of propositional variables
$\mathrm{Var}(\Phi)$	Set of propositional variables occurring in the formulas in set Φ
$\mathbf{X}_n$	Set of propositional variables $\{X_1, \ldots, X_n\}$
$\mathbf{Val}^{\mathcal{P}}$	A semantics of partial valuations over propositional variables in set $\mathcal{P}$
$\mathbf{Val}^{\mathcal{P}}_\Phi$	Semantics $\mathbf{Val}^{\mathcal{P}}$ modulo satisfiability of the set Φ
$\mathbf{L}$-SAT	Satisfiability problem over logical system $\mathbf{L}$
CPL	The logical system of Classical Propositional Logic
Ł_∞	The logical system of Łukasiewicz Infinitely-valued Logic
$\vee$	CPL-disjunction or Ł_∞-maximum
$\wedge$	CPL-conjunction or Ł_∞-minimum
$\neg$	CPL- or Ł_∞-negation
$\rightarrow$	CPL- or Ł_∞-implication
$\leftrightarrow$	CPL- or Ł_∞-bi-implication
$\oplus$	Ł_∞-disjunction
$\odot$	Ł_∞-conjunction
nX	Ł_∞-disjunctions of n-fold repetitions of the propositional variable X
$\mathbf{0}$	Defined formula in Ł_∞ with constant value 0
$\mathbf{1}$	Defined formula in Ł_∞ with constant value 1
PSAT	Probabilistic Satisfiability Problem over CPL
Ω°	Topological interior of set Ω
$\mathrm{cl}(\Omega)$	Topological closure of set Ω
$\partial\Omega$	Topological boundary of set Ω
$\mathrm{conv}(\Omega)$	Convex hull of set Ω
P	The complexity class of polynomial problems
NP	The complexity class of nondeterministic polynomial problems

$Ł_\infty$-MODSAT	System for function representation modulo satisfiability
ŁIPSAT	Probabilistic Satisfiability Problem over $Ł_\infty$
PCE-Coherence	The problem of deciding coherence of a set of PCE
PCE-Extension	The problem of computing maximum or minimum values of a probabilistic assignment to an action
GNP_k^s	GNP-class of games whose players have at most s actions and k neighbors
GNP_k	GNP-class of games whose players have at most k neighbors
2GNP	GNP-class of 2-player games allowing mixed equilibria
2G	Class of 2-player games allowing mixed equilibria
$\|A\|$	Cardinality of set A
A^*	Number set A without number 0
A_+	Number set A without negative numbers
$\lfloor x \rfloor$	Floor of number x
$\lceil x \rceil$	Ceil of number x
x_2	Number x written in binary notation
$n!$	Factorial of number n
$\log n$	Logarithm base 2 of number n
$O(f(n))$	Big O notation applied to function $f(n)$

List of Figures

List of Tables

Chapter 1

Introduction

Applications of logical systems often explore the evaluations of logical formulas by means of a valuation semantics. However, in this thesis we investigate a restricted evaluation of formulas that takes into account only a part of the whole set of valuations, which is constrained to the satisfiability of a specific set of formulas; as the set of all valuations is called the semantics of the logical system, we say that such kind of restricted evaluation takes place in a *semantics modulo satisfiability*. The broader objective of this work is to identify and develop computationally efficient applications of semantics modulo satisfiability; e.g. in the representation of piecewise linear functions into Łukasiewicz Infinitely-valued Logic, which we further apply to the formal analysis of neural networks.

First considering standard semantic evaluation, on the one hand, valid formulas or tautologies — i.e. formulas satisfied by every valuation — are some of the main concerns in the study of logical systems; on the other hand, non-valid formulas may play important roles in contexts where it is also necessary for their evaluations to take non-designated truth values; for instance, such formulas may represent functions into a logical system. This is the case of the known property of formulas of Classical Propositional Logic that represent Boolean functions; this is because semantic evaluation of these representative formulas have the property of matching the values of Boolean functions when their propositional variables are associated with the function variables. There are formal verification techniques that depend on the representation of Boolean functions into logic.

The standard semantic evaluation also arises on probability assignment to logical formulas. When grounding a probability theory on a logical system — both classical and non-classical —, it is usual for probability distributions over its semantics to induce probabilities to formulas in the language of the system. In this way, the probability of a formula may be computed as the probabilistic average of the evaluations of such formula weighed by the probability distribution over the semantics. Of course, non-valid formulas may have nonzero probability, which makes their probability computation to consider evaluations with non-designated truth values.

Together with valid formulas, logical consequence — i.e. the relation between a set of formulas called premises and another formula called conclusion which is satisfied by all valuations that also satisfy the premises — is a leading concept in logical systems. Indeed, a logical system may be seen as a pair comprehending a language and a logical consequence relation; fixing a set of formulas called axioms, a logical theory of all the formulas which are logical consequences from those axioms is determined. Analogous to non-valid formulas, we claim that non-logical consequences are also important for the semantic properties of non-conclusions when evaluated according to valuations that satisfy a set of premises. Properly used, non-logical consequences may lead to

efficient computational treatment of many problems. Such evaluation of formulas in accordance with a set of premises or a logical theory is done within a semantics modulo satisfiability.

We may identify an algorithmic use of semantics modulo satisfiability by Finger and De Bona (2015) in the context of the Probabilistic Satisfiability Problem, which consists in attesting the coherence of probabilistic assignments to formulas within the framework of Classical Propositional Logic, that is to check the existence of a probability distribution over the semantics that induces the assignment under concern. State-of-the-art solvers for such problem focus on specific instances — in the so-called atomic normal form — where only maximum probability may be assigned to non-atomic formulas; such assignment forces the probability distribution over semantics to only assign nonzero probabilities to valuations that satisfy these non-atomic formulas. Thus, the computation of the probabilities of the atomic formulas must take into account their values only according to valuations satisfying the non-atomic ones. As a consequence, the mentioned solvers operate by searching for valuations in a semantics modulo satisfiability.

In this work, we tackle problems about function representation, probabilistic satisfiability and game theory by means of semantics modulo satisfiability. Let us take an overview of them.

In contrast to the direct representation of Boolean functions into Classical Propositional Logic, we present a way to implicitly represent continuous piecewise linear functions into Łukasiewicz Infinitely-valued Logic — the representation modulo satisfiability — which is performed by evaluating the representative formula within a semantics modulo satisfiability. As continuous piecewise linear functions may approximate any continuous function — by Weierstrass-like approximations —, representing them in some logical system is a preparation step for the application of formal verification and automated reasoning techniques to the study of systems modeled by these functions. Then, we apply such representation system to formally verify neural networks that compute piecewise linear functions regarding the properties of reachability and robustness.

Continuous piecewise linear functions may be directly represented in many logical systems. However, our approach brings together some features which we are unaware to have been jointly considered in the existing literature.

- There is a polynomial algorithm that generates the representation of continuous piecewise linear functions given in a suitable encoding.
- It takes place in Łukasiewicz Infinitely-valued Logic, which is a logical system whose main associated computational problems are classified in reasonable complexity classes; for instance, satisfiability is "only" NP-complete (Mundici, 1987).
- There already exists extensive literature on solvers for Łukasiewicz Infinitely-valued Logic and, in a practical view, some solvers have been tested and the empirical phase transition phenomenon has been identified in them (Bofill *et al.*, 2015).

In another application of semantics modulo satisfiability, we study the problem of deciding coherence of probabilistic assignments to formulas of Łukasiewicz Infinitely-valued Logic; such problem has been theoretically studied and shown to be NP-complete by Bova and Flaminio (2010). However, a deterministic algorithm for solving it was still missing; thus, in analogy to the aforementioned algorithm for the Probabilistic Satisfiability Problem, we establish an atomic normal form of probabilistic assignments which leads to an algorithm that works by searching valuations in a semantics

modulo satisfiability. We also present the empirical detection of the phase transition phenomenon in an implementation of such algorithm.

At last, we propose and study some problems in game theory related to observable games, which are problems that arise when a game reaches a Nash equilibrium, however there is uncertainty about which exact equilibrium this is; such uncertainty is put in terms of probabilistic assignments to the actions the players may perform. We model this scenario by probabilistic assignments to atomic formulas that may only be computed from probability distributions over valuations representing Nash equilibria, which, in turn, are valuations in a specific semantics modulo satisfiability. Two settings are considered: first, allowing only pure Nash equilibria, which are modeled over Classical Propositional Logic; second, also allowing mixed Nash equilibria, which are modeled over Łukasiewicz Infinitely-valued Logic. Thus, we are able to establish the decision problem of coherence of the probabilistic assignments to actions as NP-complete and provide algorithms for it by means of reductions to problems of coherence of probabilistic assignments to formulas in an adequate logic. From the decision problem, we also derive algorithms for the Extension Problem, which is that of computing the maximum and minimum values an unconstrained action may take in an already coherent setting.

1.1 Publications

Part of this thesis' results, as well as related research done by the author during his doctoral studies, has appeared in workshops, conferences and journals.

- Preto and Finger (2020) Sandro Preto and Marcelo Finger. An efficient algorithm for representing piecewise linear functions into logic. *Electronic Notes in Theoretical Computer Science*, 351:167-186. ISSN 1571-0661. doi: 10.1016/j.entcs.2020.08.009. URL http://doi.org/10.1016/j.entcs.2020.08.009. Proceedings of LSFA 2020, the 15th International Workshop on Logical and Semantic Frameworks, with Applications (LSFA 2020).

- Finger and Preto (2020) Marcelo Finger and Sandro Preto. Probably partially true: Satisfiability for Łukasiewicz infinitely-valued probabilistic logic and related topics. *Journal of Automated Reasoning*, 64(7):1269-1286. ISSN 1573-0670. doi: 10.1007/s10817-020-09558-9. URL http://doi.org/10.1007/s10817-020-09558-9.

- Preto and Finger (2019) Sandro Preto and Marcelo Finger. Representing rational McNaughton functions via MODSAT relativisation. In Cezar Augusto Mortari, Ricardo Silvestre, Ítala Maria Loffredo D'Ottaviano, Leandro Suguitani and Petrucio Viana, editors, *19th Brazilian Logic Conference EBL 2019: Book of Abstracts*, page 183. Mídia Gráfica e Editora Ltda, UFCG-EDUFCG.

- Salvatore *et al.* (2019) Felipe Salvatore, Sandro Preto, Marcelo Finger and Roberto Hirata Jr. Using neural models to perform inference. In Derek Doran, Artur d'Avila Garcez and Freddy Lecue, editors, *Proceedings of the 2019 International Workshop on Neural-Symbolic Learning and Reasoning*.

- Finger and Preto (2018) Marcelo Finger and Sandro Preto. Probably half true: Probabilistic satisfiability over Łukasiewicz infinitely-valued logic. In Didier Galmiche, Stephan Schulz and Roberto Sebastiani, editors, *Automated Reasoning. IJCAR 2018*, volume 10900 of *Lecture Notes in Computer Science*, pages 194-210, Cham. Springer International Publishing. ISBN 978-3-319-94205-6.

Furthermore, the following papers, which are also related to this thesis, have been submitted to journals and are currently under review.

- Sandro Preto and Marcelo Finger. Efficient representation of piecewise linear functions into Łukasiewicz logic modulo satisfiability.
- Sandro Preto and Marcelo Finger. Proving properties of binary classification neural networks via Łukasiewicz logic.
- Sandro Preto, Eduardo Fermé and Marcelo Finger. Coherence of probabilistic constraints on Nash equilibria.

1.2 Thesis Structure

In Chapter 2 we introduce the central concept of semantics modulo satisfiability together with all reasoning tools that are used in the thesis: general logical systems, two propositional logics — Classical Propositional Logic and Łukasiewicz Infinitely-valued Logic — and classical probability theory. We also include notation and references for the auxiliary mathematical subjects that are needed for approaching the problems: topology, convex geometry, linear programming and computational complexity. Then, the next three chapters deal with the main contributions of this work.

Chapter 3 presents the technique of representation modulo satisfiability, which is used to efficiently represent, in an implicit way, a class of continuous piecewise linear functions — the rational McNaughton functions — in Łukasiewicz Infinitely-valued Logic. It also discusses an implementation of the representation building routine and an application to the formal analysis of neural networks.

Chapter 4 studies the probabilistic reasoning over Łukasiewicz Infinitely-valued Logic; it brings a theoretical investigation with the proposal of the atomic normal form as input format for the decision problem of the satisfiability of probabilistic assignments. A solving algorithm, its implementation and discussion are also introduced.

Chapter 5 initially deviates from the logical and probabilistic settings and proposes problems in game theory related to observable games. Such problems are approached via reductions to the previously addressed problems of satisfiability of probabilistic assignments to logical formulas. The chapter also has complexity analyses and proposals of algorithms for the studied problems.

In Chapter 6, we draw some conclusions about the use of semantics modulo satisfiability in general, the particular matters treated in the central chapters and the use of semantics modulo satisfiability in those particular matters. We also propose some possible future work departing from this thesis.

Chapter 2

Preliminaries

Semantics modulo satisfiability underlie all the techniques we propose in this work; thus, first we define this concept encompassed in a general framework of logical systems in Section 2.1. Then, we introduce propositional logics and, in particular, the systems of Classical Propositional Logic and Łukasiewicz Infinitely-valued Logic in Section 2.2. In Section 2.3, we introduce classical probabilities defined over Classical Propositional Logic. Finally, we give references for other mathematical subjects that underlie this work in Section 2.4.

2.1 Logic and Semantics Modulo Satisfiability

A *logical system*, or simply a *logic*, is a pair $\mathbf{L} = \langle \mathcal{L}, \models \rangle$, where $\mathcal{L}$ is a set of *formulas*, called the *language* of the system, and $\models$ is a *(logical) consequence relation* in $\wp(\mathcal{L}) \times \mathcal{L}$; if $\Gamma \models \varphi$ holds, where $\Gamma \subseteq \mathcal{L}$ and $\varphi \in \mathcal{L}$, the formulas in Γ are the *premises* and φ is the *conclusion* of the logical consequence. In this work we will only consider logical systems defined through valuation semantics; thus, besides a language $\mathcal{L}$, we need a set of *truth values* $\mathcal{V}$ and a set of *designated truth values* $\mathcal{D} \subseteq \mathcal{V}$ among them. A *(valuation) semantics* for the logical system is a set of valuations denoted by **Val**; a *valuation* is a function $v : \mathcal{L} \to \mathcal{V}$ that assigns a value in $\mathcal{V}$ to each formula in $\mathcal{L}$; to evaluate a formula $\varphi \in \mathcal{L}$ according to a valuation v means to compute the value $v(\varphi)$.

We say that a valuation $v \in \mathbf{Val}$ *satisfies* a formula $\varphi \in \mathcal{L}$ if $v(\varphi) \in \mathcal{D}$ and that it *satisfies* a set of formulas $\Gamma \subseteq \mathcal{L}$ if it satisfies all formulas $\varphi \in \Gamma$; we denote both cases by $v \models \varphi$ and $v \models \Gamma$, respectively. A formula or a set of formulas is *satisfiable* if there is some valuation that satisfies it; otherwise, it is *unsatisfiable*. Now, we may define the consequence relation and write, for $\varphi \in \mathcal{L}$ and $\Gamma \subseteq \mathcal{L}$,

$$\Gamma \models \varphi,$$

if, for every valuation $v \in \mathbf{Val}$ such that $v \models \Gamma$, we also have that $v \models \varphi$; φ is said to be a *logical consequence* from Γ. A *valid formula* or *tautology* is a formula φ which is a logical consequence of the empty set; we write

$$\models \varphi$$

instead of $\varnothing \models \varphi$ and note that a formula φ is valid if, and only if, $v \models \varphi$, for all $v \in \mathbf{Val}$.

In a logical system defined through valuation semantics, tautologies have a prominent place due to their property of being always true. Nevertheless, non-valid formulas may be of great interest for their values according to the semantics. We give some examples.

- By the Cook-Levin Theorem (Cook, 1971), the decision problems belonging to the complexity class NP — which constitute several of the problems of practical interest (see Section 2.4 for references) — have instances that may be translated into formulas of the Classical Propositional Logic, which are **Yes**-instances, if there is a valuation that satisfies its associated formula, or **No**-instances, otherwise. Such formulas are not tautologies in general and the semantics of Classical Propositional Logic provide a proof system for the **Yes**-instances.

- By means of their possible evaluations according to the semantics, formulas may represent (multivariate) functions which have arguments and take values in the set of truth values $\mathcal{V}$. In general, such formulas are also not tautologies and they may be part of other larger formulas that state properties about the represented functions; in turn, for such larger formulas, one might be concerned whether it is a tautology. This is the case, for instance, of Boolean functions, which may be represented by formulas of Classical Propositional Logic and might model the behavior of electronic circuits; in this way, Classical Propositional Logic reasoning may be used to verify whether such circuits are in accordance with specifications also codified in the language of such logical system (McFarland, 1993).

- A non-valid formula does not necessarily stand for an impossibility, however, one might not be sure whether such formula is true or not in some context. Thus, it is natural to assign a probability value to such formula. Assigning probabilities to formulas of a logical system gives rise to a probability theory and such assignments are expected to be in agreement with the (not necessarily designated) evaluations of such formulas in the system semantics (see Section 2.3 for a classical probability theory).

Together with tautologies, logical consequences have central role in the study of logic. In this way, we define a *logical theory* determined by a set of formulas $\Gamma \subseteq \mathcal{L}$, called *axioms*, with $\langle \mathcal{L}, \models \rangle$ as underlying logical system, by the set

$$\mathrm{Th}(\Gamma) = \Big\{ \varphi \in \mathcal{L} \;\Big|\; \Gamma \models \varphi \Big\}.$$

Likewise our considerations about non-valid formulas, we claim that, in the context of logical theories, there might be interest in formulas which are not logical consequences from the axioms. Of course, the values of such (non-logical consequence) formulas must be in accordance with the theory. Thereby, they only should be evaluated by valuations that satisfy the axioms, which leads to a constraining to the semantics of the logical system in question. With this motivation, we establish semantics modulo satisfiability as a part of the semantics of a logical system comprehending the valuations constrained to the satisfiability of some set of formulas.

Definition 1 Let $\Phi \subseteq \mathcal{L}$ be a set of formulas, a *semantics modulo satisfiability* is the set

$$\mathbf{Val}_\Phi = \Big\{ v \in \mathbf{Val} \;\Big|\; v \models \Phi \Big\}.$$

If set $\Phi = \{\varphi\}$ is a singleton, we write $\mathbf{Val}_\varphi$. □

In a model-theoretic point of view, a semantics modulo satisfiability $\mathbf{Val}_\Phi$ is the set of the *models* of the logical theory $\mathrm{Th}(\Phi)$; model theory focuses on the relations between logical theories and their models, so formulas satisfied by models are the ones primarily regarded in such context.

On the other hand, our main goal in this work is to exploit the evaluations of formulas in general through semantics modulo satisfiability in order to obtain efficient computational performance in diversified tasks.

All the concepts related to a logical system $\langle \mathcal{L}, \models \rangle$ with semantics $\mathbf{Val}$ defined in this section may be reformulated in terms of semantics modulo satisfiability by the following equivalent definitions.

- A formula φ is satisfiable if $\mathbf{Val}_\varphi \neq \varnothing$.
- A logical consequence $\Gamma \models \varphi$ holds if $\mathbf{Val}_\Gamma \subseteq \mathbf{Val}_\varphi$.
- A formula φ is valid if $\mathbf{Val} = \mathbf{Val}_\varphi$.
- A logical theory $\mathrm{Th}(\Gamma)$ is the set $\{\varphi \in \mathcal{L} \mid \mathbf{Val}_\Gamma \subseteq \mathbf{Val}_\varphi\}$.

These equivalent definitions show that constraining a semantics is not an alien procedure, on the contrary, it is implicit in many traditional logical concepts. The novelty in this work is the unusual importance given to the evaluation of formulas which assume any truth value, and not necessarily designated ones, by constrained valuations in a semantics modulo satisfiability with the aim to achieve efficiency gains in computational tasks.

2.2 Propositional Logics

In this work, we employ logical systems $\mathbf{L} = \langle \mathcal{L}, \models \rangle$ which are *propositional logics* and whose languages are generated from a countable set of *propositional variables* or *atoms* $\mathbb{P}$. The formulas in $\mathcal{L}$ are finite sequences of symbols which may be propositional variables in $\mathbb{P}$, a *unary operator* $\Box$, a *binary operator* $\Diamond$ or parentheses. Let $\mathcal{L}_0 = \mathbb{P}$ and

$$\mathcal{L}_i = \mathcal{L}_{i-1} \ \cup \ \Big\{ \Box\varphi \ \Big| \ \varphi \in \mathcal{L}_{i-1} \Big\} \ \cup \ \Big\{ (\varphi \Diamond \psi) \ \Big| \ \varphi, \psi \in \mathcal{L}_{i-1} \Big\},$$

for $i = \mathbb{N}^*$; then, the propositional language is given by

$$\mathcal{L} = \bigcup_{i \in \mathbb{N}} \mathcal{L}_i.$$

Moreover, $\mathcal{L}$ is said to be *freely generated* from $\mathbb{P}$ by the operators $\Box$ and $\Diamond$ because the functions $F_\Box : \mathcal{L} \to \mathcal{L}$, given by $F_\Box(\varphi) = \Box\varphi$, and $F_\Diamond : \mathcal{L} \times \mathcal{L} \to \mathcal{L}$, given by $F_\Diamond(\varphi, \psi) = (\varphi \Diamond \psi)$, have ranges disjoint from each other and from $\mathbb{P}$ and they are both one-to-one.

All operators in the logical systems employed in this work are *truth-functional*, that means that there are functions $f_\Box : \mathcal{V} \to \mathcal{V}$ and $f_\Diamond : \mathcal{V} \times \mathcal{V} \to \mathcal{V}$ associated to the unary and binary operators $\Box$ and $\Diamond$ such that, for a valuation $v \in \mathbf{Val}$ and formulas $\varphi, \psi \in \mathcal{L}$, we have:

$$v(\Box\varphi) = f_\Box(v(\varphi)); \tag{2.1}$$

$$v(\varphi \Diamond \psi) = f_\Diamond(v(\varphi), v(\psi)). \tag{2.2}$$

Thus, one may just give a function $v_\mathbb{P}$ which maps propositional variables to a truth value in $\mathcal{V}$ and extend this function to a valuation by obeying (2.1) and (2.2); because $\mathcal{L}$ is freely generated from $\mathbb{P}$ by $\Box$ and $\Diamond$, this extension is uniquely defined by such assignment to the variables in $\mathbb{P}$ given by

$v_{\mathbb{P}}$. For a detailed treatment of the constructions so far in this section, we refer the reader to the book of Enderton (2001).

We might define other operator symbols as abbreviations in terms of the basic unary and binary ones. Also, we often omit parentheses in formulas (with or without abbreviations) when there is no danger of ambiguity according to the following conventions:

- The outermost parentheses are omitted.
- When a binary operator is used multiple times, grouping is to the right.

Operator symbols defined in terms of the basic ones are obviously also truth-functional; so, note that probabilities are not operators.

We denote propositional variables mostly by Latin capital letters with occasional subscripts or superscripts — e.g. X, X_1, X_2, Y, Z, $Z_2^{p_1}$ — and sometimes by the Greek lowercase letter ξ also with subscripts or superscripts. Let $\Phi \subseteq \mathcal{L}$ be a set of formulas; we denote by $\mathrm{Var}(\Phi)$ the set of all propositional variables that occur in all formulas $\psi \in \Phi$; if $\Phi = \{\varphi\}$ is a singleton, we denote such set by $\mathrm{Var}(\varphi)$. We also use the notation $\mathbf{X}_n = \{X_1, \ldots, X_n\}$ for such set of propositional variables.

In practical situations, it is usual to only consider some particular set of formulas $\Phi \subseteq \mathcal{L}$ such that $\mathrm{Var}(\Phi) \subseteq \mathcal{P} \subseteq \mathbb{P}$, where $\mathcal{P}$ is a finite set of propositional variables. Therefore, in such situations, we are only concerned with the values that a valuation in **Val** assigns to the propositional variables in $\mathcal{P}$; due to truth-functionality of operators, from these values we easily determine the value of a formula φ such that $\mathrm{Var}(\varphi) \subseteq \mathcal{P}$. We call *partial valuations (over $\mathcal{P}$)* those functions that assign truth values in $\mathcal{V}$ to propositional variables in $\mathcal{P} \subseteq \mathbb{P}$ and to formulas φ with $\mathrm{Var}(\varphi) \subseteq \mathcal{P}$; let us denote by $\mathbf{Val}^{\mathcal{P}}$ the set of all such partial valuations. Partial valuations over $\mathcal{P}$ may be seen as the valuations of the logical system

$$\Big\langle \{\varphi \in \mathcal{L} \mid \mathrm{Var}(\varphi) \subseteq \mathcal{P}\} \ , \ \{\langle \Gamma, \varphi\rangle \mid \Gamma \models \varphi \text{ and } \mathrm{Var}(\Gamma) \cup \mathrm{Var}(\varphi) \subseteq \mathcal{P}\}\Big\rangle.$$

In this way, $\mathbf{Val}^{\mathcal{P}}$ is a semantics and we have semantics modulo satisfiability in total analogy to the previous section denoted by $\mathbf{Val}^{\mathcal{P}}_{\Phi}$, where $\Phi \subseteq \mathcal{L}$ is a set of formulas such that $\mathrm{Var}(\Phi) \subseteq \mathcal{P}$.

Note that any single partial valuation $v \in \mathbf{Val}^{\mathcal{P}}$ may be seen as the restriction of infinitely many valuations $\tilde{v} \in \mathbf{Val}$ to the set of formulas $\{\varphi \in \mathcal{L} \mid \mathrm{Var}(\varphi) \subseteq \mathcal{P}\} \subseteq \mathcal{L}$. Thus, a partial valuation over $\mathcal{P} \subseteq \mathbb{P}$ may be extended to infinitely many distinct valuations in **Val**. Let $\varphi \in \mathcal{L}$ and $\Gamma \subseteq \mathcal{L}$ be such that $\mathrm{Var}(\varphi) \cup \mathrm{Var}(\Gamma) \subseteq \mathcal{P}$; some properties of these formulas defined by means of **Val** have equivalents only in terms of $\mathbf{Val}^{\mathcal{P}}$.

- The formula φ is satisfiable if, and only if, there is a valuation $v \in \mathbf{Val}^{\mathcal{P}}$ such that $v(\varphi) \in \mathcal{D}$.
- The logical consequence $\Gamma \models \varphi$ holds if, and only if, for any valuation $v \in \mathbf{Val}^{\mathcal{P}}$ such that $v(\gamma) \in \mathcal{D}$, for all $\gamma \in \Gamma$, it is also the case that $v(\varphi) \in \mathcal{D}$.

The *Satisfiability Problem* — denoted SAT — is the problem of deciding whether a given formula $\varphi \in \mathcal{L}$ is satisfiable; that is whether there is a valuation $\tilde{v} \in \mathbf{Val}$ such that $\tilde{v}(\varphi) \in \mathcal{D}$ or, equivalently, whether there is a valuation $v \in \mathbf{Val}^{\mathcal{P}}$ such that $\mathrm{Var}(\varphi) \subseteq \mathcal{P}$ and $v(\varphi) \in \mathcal{D}$. We might indifferently refer to the Satisfiability Problem as the problem of deciding if a set of formulas $\Phi \subseteq \mathcal{L}$ is satisfiable and we write **L**-SAT when referring to the Satisfiability Problem of a specific logic **L**. Also, we

call **L**-solver a routine that computes a satisfying partial valuation for an instance of **L**-SAT or, alternatively, states that such an instance is unsatisfiable.

2.2.1 Classical Propositional Logic

Classical Propositional Logic, denoted by CPL $= \langle \mathcal{L}_{\mathrm{CPL}}, \models_{\mathrm{CPL}} \rangle$ is probably the most known logical system. It has a 2-valued semantics with the set of truth values $\mathcal{V}_{\mathrm{CPL}} = \{0, 1\}$ and only one designated truth value in the set $\mathcal{D}_{\mathrm{CPL}} = \{1\}$. CPL-SAT was the first problem shown to be NP-complete independently by Cook (1971) and Levin (1973).

The basic CPL-language $\mathcal{L}_{\mathrm{CPL}}$ is freely generated from the countable set of propositional variables $\mathbb{P}$ by the unary *negation* $\neg$ and the binary *disjunction* $\vee$ CPL-operators. For the semantics CPL-**Val**, define a CPL-valuation as a function $v : \mathcal{L}_{\mathrm{CPL}} \to \{0, 1\}$, such that, for $\varphi, \psi \in \mathcal{L}_{\mathrm{CPL}}$:

$$v(\neg\varphi) = 1 - v(\varphi);$$
$$v(\varphi \vee \psi) = \max(v(\varphi), v(\psi)).$$

From the basic CPL-operators we derive the following ones:

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

In order to omit parentheses, we add to the already established conventions an order of operators where $\neg$ has precedence over $\vee$ and $\wedge$, which have precedence over $\to$ and $\leftrightarrow$.

CPL-solvers are usually designed to have as input CPL-formulas in *conjunctive normal form* (CNF), that is CPL-formulas in the format

$$C_1 \wedge \cdots \wedge C_n,$$

where each C_i, for $i = 1, \ldots, n$, is a *clause* in the format

$$l_1 \vee \cdots \vee l_{k_i},$$

where l_j, for $j = 1, \ldots, k_i$, is a *literal*, that is either a negated ($\neg X$) or a non-negated (X) propositional variable $X \in \mathbb{P}$. Deciding the satisfiability of the set of clauses $\{C_i\}$ is equivalent to deciding the satisfiability of the CNF CPL-formula $\bigwedge C_i$. A CPL-formula may be translated in polynomial time into a CNF CPL-formula which is satisfiable if, and only if, the original one also is.

2.2.2 Łukasiewicz Infinitely-Valued Logic

Łukasiewicz Infinitely-valued Logic, denoted by $\mathrm{Ł}_\infty = \langle \mathcal{L}_{\mathrm{Ł}_\infty}, \models_{\mathrm{Ł}_\infty} \rangle$, is arguably one of the best studied many-valued logics (Cignoli *et al.*, 2000). Being a many-valued logic means that the set of truth values of this system $\mathcal{V}_{\mathrm{Ł}_\infty} = [0, 1]$ has more than two elements; however, the set of designated truth values $\mathcal{D}_{\mathrm{Ł}_\infty} = \{1\}$ is as the one of CPL.

Łukasiewicz Infinitely-valued Logic is amenable to computational treatment; for instance, $\mathrm{Ł}_\infty$-SAT is NP-complete (Mundici, 1987), which is a reasonable complexity for a Satisfiability Problem.

Thus, it is widely used in the literature to model situations that require the notion of "partial truth"[1]; the setting studied in Chapter 4 being an example.

The basic Ł$_\infty$-language $\mathcal{L}_{Ł_\infty}$ is freely generated from the countable set of propositional variables $\mathbb{P}$ by the unary *negation* $\neg$ and the binary *disjunction* $\oplus$ Ł$_\infty$-operators. For the semantics Ł$_\infty$-**Val**, define a Ł$_\infty$-valuation as a function $v : \mathcal{L}_{Ł_\infty} \to [0,1]$, such that, for $\varphi, \psi \in \mathcal{L}_{Ł_\infty}$:

$$v(\neg\varphi) = 1 - v(\varphi); \tag{2.3}$$

$$v(\varphi \oplus \psi) = \min(1, v(\varphi) + v(\psi)). \tag{2.4}$$

From the basic Ł$_\infty$-operators we derive the following ones:

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

In order to omit parentheses, we add to the already established conventions an order of operators where $\neg$ has precedence over $\oplus$ and $\odot$, which have precedence over $\vee$ and $\wedge$, which have precedence over $\to$ and $\leftrightarrow$.

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 $v \in$ Ł$_\infty$-**Val**; we define the constant $\mathbf{0}$ by $X \odot \neg X$ and the constant $\mathbf{1}$ by $\neg\mathbf{0}$. We also define $0\varphi =_{\text{def}} \mathbf{0}$, $n\varphi =_{\text{def}} \varphi \oplus \cdots \oplus \varphi$, n times, for $n \in \mathbb{N}^*$, and $\bigoplus_{i\in\varnothing} \varphi_i =_{\text{def}} \mathbf{0}$.

Ł$_\infty$ is said to have CPL as limit case because regarding only truth values 0 and 1, the Ł$_\infty$-operators behave just like their analogous CPL-operators denoted with same symbols. Moreover, Ł$_\infty$-operators $\oplus$ and $\odot$ behave as CPL-operators $\vee$ and $\wedge$, respectively.

2.3 Classical Probability Theory

Probability theory may be grounded in a logical system by assigning probabilities to the formulas of such system in a way that takes into account its underlying logical structure. The original formulation of such a mix of Classical Propositional Logic and (discrete) probabilities is due to George Boole who, in his seminal work introducing what is now known as Boolean algebras, already discussed the problem (Boole, 1854).

Probability values assigned to a CPL-formula α, such that $\text{Var}(\alpha) \subseteq \mathcal{P} \subseteq \mathbb{P}$, for some finite set $\mathcal{P}$, are induced by a *probability distribution* $\pi :$ CPL-$\mathbf{Val}^{\mathcal{P}} \to [0,1]$ over the $2^{|\mathcal{P}|}$ CPL-partial valuations in CPL-$\mathbf{Val}^{\mathcal{P}}$ that maps every CPL-partial valuation to a probability in the real interval $[0,1]$ in a way that

$$\sum_{v\in\text{CPL-}\mathbf{Val}^{\mathcal{P}}} \pi(v) = 1.$$

[1] By the term "partial truth" we refer to the concept usually referred in the literature as "degree of truth", not to be confused with partial valuations or models.

The probability of a CPL-formula α, with $\mathrm{Var}(\alpha) \subseteq \mathcal{P}$, according to probability distribution π is given by

$$P_\pi(\alpha) = \sum \Big\{ \pi(v) \;\Big|\; v(\alpha) = 1,\ v \in \text{CPL-}\mathbf{Val}^{\mathcal{P}} \Big\}.$$

The problem of deciding whether there is a probability distribution π that jointly satisfies given probabilistic assignments to formulas is called *Probabilistic Satisfiability Problem* — denoted PSAT. PSAT has been extensively discussed in the literature (Georgakopoulos *et al.*, 1988; Hansen and Jaumard, 2000; Nilsson, 1986) and has recently received a lot of attention due to the improvements in CPL-solving and linear programming techniques, having generated a variety of algorithms, for which the empirical phenomenon of phase transition (see Section 2.4 for references) is by now established (Finger and Bona, 2011; Finger and De Bona, 2015).

A *PSAT instance* is an expression of the form

$$\Sigma = \Big\{ P(\alpha_i) \bowtie_i p_i \;\Big|\; p_i \in [0,1] \cap \mathbb{Q},\ 1 \leq i \leq K \Big\},$$

where $\alpha_1, \ldots, \alpha_k$ are CPL-formulas for which $\mathrm{Var}(\alpha_i) \subseteq \mathcal{P}$, for $i = 1, \ldots, K$, where $|\mathcal{P}| = n \in \mathbb{N}^*$, and which are restricted by probability assignments $P(\alpha_i) \bowtie_i p_i$, $\bowtie_i \in \{=, \leq, \geq\}$, for $i = 1, \ldots, K$. The Probabilistic Satisfiability Problem consists in determining if that set of constraints Σ is consistent.

A linear algebraic formulation of PSAT is provided by Nilsson (1986), it consists of a $K \times 2^n$ matrix $A = [a_{ij}]$ such that $a_{ij} = v_j(\alpha_i)$. The Probabilistic Satisfiability Problem is, thus, to decide if there is a probability vector π of dimension 2^n that obeys the *PSAT restriction*:

$$\begin{aligned} A\pi &\bowtie p \\ \textstyle\sum \pi_j &= 1 \\ \pi &\geq 0 \end{aligned} \tag{2.5}$$

If there is a probability distribution π that solves (2.5), we say π satisfies Σ. In such a setting, we define a PSAT instance Σ as *satisfiable* if (2.5) is such that there is a π that satisfies it. Clearly, the conditions in (2.5) ensure π is a probability distribution. Usually the first two conditions of (2.5) are joined, A is a $(k+1) \times 2^n$ matrix with 1's at its last line, $p_1 = k+1$ in vector $p_{(k+1)\times 1}$, so $\bowtie_{k+1}$-relation is "=".

We often use in this work the following version of Carathéodory's Theorem (Brøndsted, 1983) to prove that satisfiable (or coherent) probabilistic assignments are induced by "small" probability distributions.

Proposition 1 (Carathéodory's Theorem) *If $\mathbf{x} \in \mathbb{R}^k$ can be written as a combination*

$$\mathbf{x} = \lambda_1 \mathbf{x}_1 + \cdots + \lambda_l \mathbf{x}_l,$$

where $\mathbf{x}_1, \ldots, \mathbf{x}_l \in \mathbb{R}^k$, $\lambda_1, \ldots, \lambda_l \in \mathbb{R}$, $\lambda_1, \ldots, \lambda_l \geq 0$ and $\lambda_1 + \cdots + \lambda_l = 1$ — that is $\mathbf{x}$ is a convex combination *of $\mathbf{x}_1, \ldots, \mathbf{x}_l$ —, then $\mathbf{x}$ can be written as a convex combination of at most $k+1$ elements among $\mathbf{x}_1, \ldots, \mathbf{x}_l$.* □

As consequence of Carathéodory's Theorem, Georgakopoulos *et al.* (1988) showed that if a PSAT instance $\Sigma = \{P(\alpha_i) \bowtie_i p_i \mid 1 \leq i \leq k\}$ has a solution π satisfying (2.5), then there is

a solution π' also satisfying (2.5) such that $\pi'_j > 0$ for at most $k+1$ elements; the remaining elements of π' are 0. The existence of such a small witness serves as an NP-certificate for a satisfiable instance, so PSAT is in NP. Furthermore, note that by making all probabilities 1 in (2.5), the problem becomes CPL-SAT, so PSAT is NP-hard. It follows that PSAT is NP-complete.

Probability theories, and classical probability theory in particular, are examples of where non-valid formulas play a nontrivial role, since they may have positive probabilities; tautologies are guaranteed to have probability 1. Analogously, let us discuss the role that non-logical consequences may have in classical probability theory when taking into account a classical propositional theory.

Let $P(\gamma) = 1$ be a satisfiable probabilistic assignment for every CPL-formula $\gamma \in \Gamma \subseteq \mathcal{L}_{CPL}$, where $\mathrm{Var}(\gamma) \subseteq \mathcal{P} \subseteq \mathbb{P}$, for all $\gamma \in \Gamma$, and $\mathcal{P}$ is a finite set; then, there must be an underlying probability distribution $\pi : \mathbf{Val}^{\mathcal{P}} \to [0,1]$ that assigns nonzero probability $\pi(v) > 0$ only to CPL-valuations $v \in \mathbf{Val}^{\mathcal{P}}_{\Gamma}$. In order to other added probabilistic assignments $P(\alpha) \bowtie p$ to maintain satisfiability of the original ones, the underlying probability distribution also must assign nonzero probabilities only to CPL-valuations $v \in \mathbf{Val}^{\mathcal{P}}_{\Gamma}$. We refer to this situation saying that the probability distribution π and the probabilistic assignments $P(\alpha) \bowtie p$ *agree* or *are in accordance* with the classical propositional theory $\mathrm{Th}(\Gamma)$. Observe that semantics modulo satisfiability is in the background for assigning probabilities to formulas in a way that agrees with a logical theory.

The state-of-the-art PSAT-solving algorithms presented by Finger and De Bona (2015) take as input instances in *atomic normal form* $\langle \Gamma, \Psi \rangle$, where Γ is a set of formulas and Ψ is a set of probabilistic assignments to atomic formulas. Let $\mathrm{Var}(\Gamma) \cup \mathrm{Var}(\Psi) \subseteq \mathcal{P}$, where $\mathcal{P}$ is a finite set; the atomic normal form instance $\langle \Gamma, \Psi \rangle$ is satisfiable if, and only if, there is a probability distribution $\pi : \mathbf{Val}^{\mathcal{P}}_{\Gamma} \to [0,1]$ that satisfies the assignments in Ψ. Thus, the mentioned algorithm works by searching for a probability distribution over the semantics modulo satisfiability $\mathbf{Val}^{\mathcal{P}}_{\Gamma}$. PSAT instances Σ may be put in atomic normal form in polynomial time and the mentioned algorithms have the empirical phenomenon of phase transition detected.

2.4 Useful Mathematical Techniques

Up to this moment, we have introduced many reasoning frameworks which are necessary either for placing or modeling the problems we approach throughout this work. Nevertheless, tackling and discussing those problems also requires some notions of topology, geometry, linear programming and computational complexity. In this section we limit ourselves to establish the notation we use and refer the reader to reference works on those subjects.

Let $\Omega \subseteq [0,1]^n$; we denote by Ω° its interior, by $\mathrm{cl}(\Omega)$ its closure and by $\partial\Omega$ its boundary in the usual topology of $[0,1]^n$. The book of Munkres (2000) is a comprehensive guide to these topological concepts as well as limits, continuous functions and open, closed, dense, connected and compact sets. We denote by $\mathrm{conv}(\Omega)$ the convex hull of $\Omega \subseteq [0,1]^n$. Such geometrical concept together with affine spaces, polyhedra and simplices and also the Carathéodory's Theorem (Proposition 1) are dealt with in the book of Brøndsted (1983).

The problem of computing the maximum of minimum value or a linear function that takes values over a polyhedron lies within the scope of linear programming. A comprehensive study of such problem that provides algorithms — as the famous simplex algorithm — and their complexity analysis may be found in the books of Bertsimas and Tsitsiklis (1997) and Papadimitriou and Steiglitz

(1998). The book of Borgward (1986) brings a probabilistic complexity analysis of the simplex method.

We denote by P the class of decision problems for which there is a (deterministic) polynomial decision algorithm and by NP the class of decision problems for which there is a nondeterministic polynomial decision algorithm. Some treatments for such classes and others as well as a theoretical approach to algorithms, decision and search problems, reductions and the NP-completeness theory are found in the books of Goldreich (2008) and Papadimitriou (1994). The phenomenon of phase transition is presented by Cheeseman *et al.* (1991); we briefly introduce it in Section 4.4.

Chapter 3

Efficient Representation of Piecewise Linear Functions into Logic

The ability to represent real continuous functions of real variables by logical formulas might allow us to apply automated reasoning techniques to the study of real systems whose behavior is modeled by these functions; however, the fact that there are uncountable many such functions frustrates the possibility to represent them in a computable formal language. This issue may be circumvented by representing the functions within an enumerable class which is dense in the class of continuous functions one desires to represent; this way, we have approximate representations of such continuous functions. It is also important for such representational ability to be effective that there exist efficient ways to generate the formulas in a target logic in which reasoning is not exceedingly complex.

In this chapter, we are concerned with the representation of rational McNaughton functions, that are continuous $[0,1]$-valued piecewise linear functions with rational coefficients over $[0,1]^n$. Rational McNaughton functions may approximate any $[0,1]$-valued continuous function over $[0,1]^n$ as stated by the following Weierstrass-like results (Aguzzoli and Mundici, 2001, 2003; Amato and Porto, 2000) and depicted in Figure 3.1.

Proposition 2 (Variation of Weierstrass Approximation Theorem) *Let $f : [0,1]^n \to [0,1]$ be a continuous function and $\varepsilon > 0$. Then there is a rational McNaughton function $\tilde{f} : [0,1]^n \to [0,1]$ such that $|f(\mathbf{x}) - \tilde{f}(\mathbf{x})| < \varepsilon$, for all $\mathbf{x} \in [0,1]^n$.* ■

We introduce an implicit kind of function representation based on semantics modulo satisfiability — the *representation modulo satisfiability* — with the aim to enlarge the representational capacity of Łukasiewicz Infinitely-valued Logic in order to also encompass rational McNaughton functions; we call the developed framework the Ł_∞-MODSAT system. For such system, we are able to provide a polynomial algorithm that builds a representation of a rational McNaughton function given in the *regional format*, an input format we determine.

Furthermore, we show how representation in Ł_∞-MODSAT combined with the possibilities of expression in Ł_∞ may be used to perform formal analysis of neural networks that compute rational McNaughton functions regarding the properties of accessibility and robustness. We also employ the introduced techniques in the analysis of a neural network trained to predict whether it will rain tomorrow in Australia.

In Section 3.1, we introduce the traditional way of representing functions in Ł_∞, in Section 3.2, we carry out a theoretical investigation of representation modulo satisfiability and, in Section 3.3,

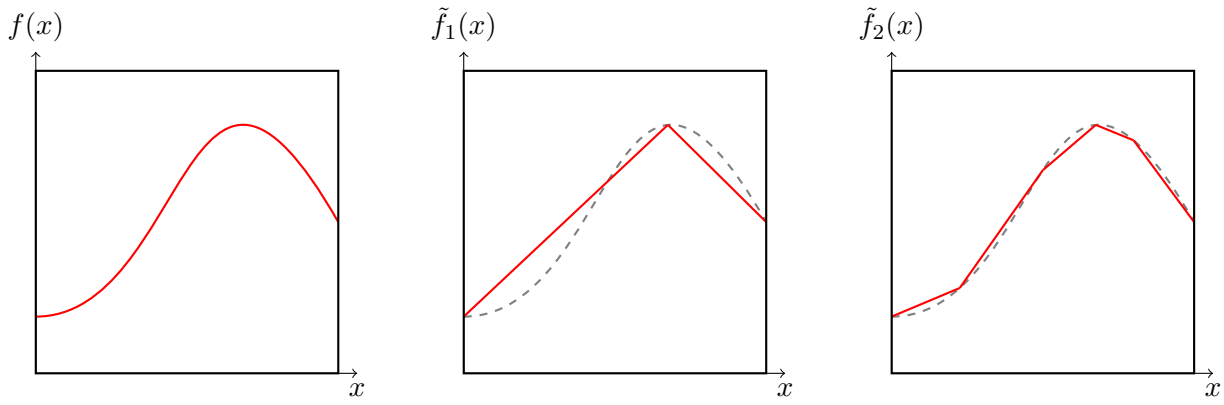


Figure 3.1: *Continuous one-variable function approximated by rational McNaughton functions*

we show that rational McNaughton functions may be represented in this way. In Section 3.4, we describe a polynomial algorithm for building representations of rational McNaughton functions in Ł$_\infty$-MODSAT and, in Section 3.5, we discuss an implementation of our algorithm and present some experimental results. In Section 3.6, we apply representation in Ł$_\infty$-MODSAT to the formal analysis of neural networks. Finally, we discuss the related work on logical representation of rational McNaughton functions in Section 3.7.

In this entire chapter, we refer to Ł$_\infty$-formulas, Ł$_\infty$-valuations, Ł$_\infty$-partial valuations and the semantics Ł$_\infty$-**Val** as formulas, valuations, partial valuations and **Val**.

3.1 The Traditional Way

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

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

$$f_\varphi(v(X_1), \ldots, v(X_n)) = v(\varphi), \text{ for any } v \in \mathbf{Val}. \tag{3.1}$$

And, then, we say that formula φ *represents* function f. A *McNaughton function* $f : [0,1]^n \to [0,1]$ is a function that satisfies the following conditions:

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

In analogy to this representational framework, the formulas of other logics may represent other classes of functions. For instance, the formulas of Classical Propositional Logic represent all the Boolean functions. Moreover, there are many logics whose formulas represent the aforementioned rational McNaughton functions, which are generalized McNaughton functions whose linear pieces coefficients may also be *rational numbers*; see Section 3.7.

Our enterprise is to develop a new way of function representation which enables rational McNaughton functions to be represented in the very Ł_∞.

3.2 Representation Modulo Satisfiability

Although formulas of Ł_∞ only represent (integer) McNaughton functions, we might use semantics modulo satisfiability in order to implicitly represent rational McNaughton functions in a way we call *representation modulo satisfiability*. By this method, a (not necessarily rational McNaughton) function f is represented by a pair $\langle \varphi, \Phi \rangle$, where φ is a formula that semantically acquires values $f(\mathbf{x})$, for $\mathbf{x} \in [0,1]^n$, from valuations in $\mathbf{Val}_\Phi$, where Φ is a set of formulas. Next, we propose two different definitions for such a representation concept, which highlight different aspects of the intended technique; then we compare both of them.

3.2.1 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 $\text{Var}(\varphi) \subseteq \mathbf{X}$. Thus, if $\mathbf{X}$ is semantically identified to the domain of a function, formula φ may semantically provide the values such function take. Let us generalize this notion.

Definition 2 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 $j = 1, \ldots, n$;
- 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 $\text{Var}(\varphi) \subseteq \mathbf{X}_n$, $\mathbf{X}_n$ determines φ modulo $\varnothing$-satisfiable, by truth-functionality and the fact that $\mathbf{Val}_\varnothing = \mathbf{Val}$. Then, representation modulo satisfiability in the formula-based approach is defined in a way that retrieves property (3.1).

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 *represents* f *modulo* Φ*-satisfiable* or that $\langle \varphi, \Phi \rangle$ *represents* f *(in the system* Ł_∞*-MODSAT)* if:

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

The definition of representation modulo satisfiability in the formula-based approach adapts the aforementioned property of formulas that makes them suitable for representing functions to the context of 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

$$\Phi = \Big\{ Z_1 \oplus Z_1 \leftrightarrow X_1, \quad Z_{\frac{1}{2}} \leftrightarrow \neg Z_{\frac{1}{2}}, \quad Z_1 \to Z_{\frac{1}{2}} \Big\}.$$

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 $Z_{\frac{1}{2}}$ 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 $f(x_1) = v(Z_1)$. □

3.2.2 The Function-Based Approach

Another definition of representation modulo satisfiability may be achieved by recognizing the implicit representation of a function inside the traditional representation of another function with constrained domain.

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

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

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 3.1:

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

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.

Definition 4 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 $\mathrm{Var}(\varphi) \cup \mathrm{Var}(\Phi) = \mathbf{X}_m$, $m \geq n$, 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 $D_{\langle \varphi, \Phi \rangle} \subseteq [0,1]^m$ 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 to 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 $Z_{\frac{1}{2}}$ by X_2 and X_3 to fit the definition, which results in $\langle X_2, \Phi \rangle$, where

$$\Phi = \Big\{ X_2 \oplus X_2 \leftrightarrow X_1, \quad X_3 \leftrightarrow \neg X_3, \quad X_2 \to X_3 \Big\}.$$

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:

- $z_1(x_1) = \frac{x_1}{2}$;
- $z_2(x_1) = \frac{1}{2}$.

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 3.2 has graphs of the functions f_φ and $f_{\langle \varphi, \Phi \rangle}$ and of the set $D_{\langle \varphi, \Phi \rangle}$. □

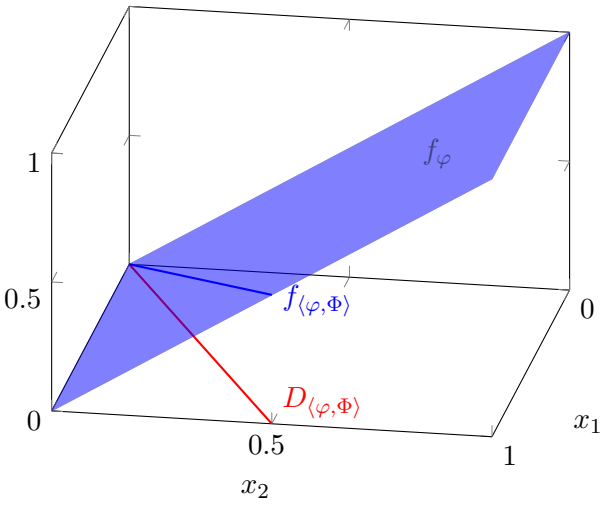


Figure 3.2: *Graphs of functions f_φ and $f_{\langle \varphi, \Phi \rangle}$ and of set $D_{\langle \varphi, \Phi \rangle}$ in Example 2, for fixed $x_3 = \frac{1}{2}$*

3.2.3 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; 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 At the same time that the pair $\langle X_2 \oplus X_3, \Phi \rangle$, where

$$\Phi = \Big\{ X_2 \leftrightarrow \neg X_3 \Big\},$$

is a representation (in the formula-based approach) for the constant function $f : [0,1] \to [0,1]$, given by $f(x_1) = 1$, it is not a functional representation for it. Note that, 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 2 the following items:

- $\mathrm{Var}(\varphi) \cup \mathrm{Var}(\Phi) = \mathbf{X}_m$, $m \geq n$;
- 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 $v(X_j) = v'(X_j)$, 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 recursive 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 2 as *strong determination modulo satisfiability* and to the consequent more restrictive form of Definition 3 as *strong representation modulo satisfiability.*

Theorem 1 *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 $z_j(\mathbf{x}) = v_{\mathbf{x}}(X_{n+j})$, where $v_{\mathbf{x}} \in \mathbf{Val}_\Phi$ is such $v_{\mathbf{x}}(X_j) = x_j$, 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 v, with $v(X_j) = x_j$, for $j = 1, \ldots, m$, we have that $v(\psi) = f_\psi(x_1, \ldots, x_m) = 1$; then $v \in \mathbf{Val}_\Phi$ and $x_{n+j} = v(X_{n+j}) = v_{\langle x_1, \ldots, x_n \rangle}(X_{n+j}) = z_j(x_1, \ldots, x_n)$, for $j = 1, \ldots, m-n$. Finally, for any $\langle x_1, \ldots, x_n \rangle \in [0,1]^n$, there is a $v \in \mathbf{Val}_\Phi$ such that $v(X_i) = x_i$, for $i = 1, \ldots, n$. Therefore, $\langle v(X_1), \ldots, v(X_m) \rangle \in D_{\langle \varphi, \Phi \rangle}$, $f(x_1, \ldots, x_n) = f(v(X_1), \ldots, v(X_n)) = v(\varphi) =$

$f_{\langle\varphi,\Phi\rangle}(v(X_1),\ldots,v(X_m)) = f(x_1,\ldots,x_n,z_1(x_1,\ldots,x_n),\ldots,z_{m-n}(x_1,\ldots,x_n))$ 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 $\psi\in\Phi$, $v(\psi) = f_\psi(\mathbf{x}, z_1(\mathbf{x}),\ldots,z_{m-n}(\mathbf{x})) = 1$, for a valuation $v\in\mathbf{Val}_\Phi$, where $v(X_1) = x_1,\ldots,v(X_n) = x_n, v(X_{n+1}) = z_1(\mathbf{x}),\ldots,v(X_m) = z_{m-n}(\mathbf{x})$. For $v, v'\in\mathbf{Val}_\Phi$, where $\mathbf{x} = \langle v(X_1),\ldots,v(X_n)\rangle = \langle v'(X_1),\ldots,v'(X_n)\rangle$, we have $\langle\mathbf{x}, v(X_{n+1}),\ldots,v(X_m)\rangle, \langle\mathbf{x}, v'(X_{n+1}),\ldots,v'(X_m)\rangle \in D_{\langle\varphi,\Phi\rangle}$, then $v(X_{n+j}) = z_j(\mathbf{x}) = v'(X_{n+j})$, 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). ■

Among all the versions of representation modulo satisfiability we have investigated, we choose to deal (from now on) with the formula-based approach as we deem it a clearer and less restrictive definition. Moreover, we will only refer to its original version presented in Section 3.2.1 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.

3.3 Representation Theorems via Hat Functions

We will establish a representation result stating that all rational McNaughton functions may be represented in Ł$_\infty$-MODSAT. For that, we adapt the proof of McNaughton's Theorem — which states that all McNaughton functions may be represented in Ł$_\infty$ — in the work of Mundici (1994). Such proof is constructive and makes use of Schauder hats, that are functions which we slightly modify into hat functions.

First, we remark a feature of representation modulo satisfiability: constants $\frac{1}{d}$, with $d\in\mathbb{N}^*$, may be represented by the pair

$$\langle\varphi,\Phi\rangle = \left\langle Z_{\frac{1}{d}}, \left\{ Z_{\frac{1}{d}} \leftrightarrow \neg(d-1)Z_{\frac{1}{d}} \right\} \right\rangle,$$

where formula φ is only one propositional variable — from now on denoted by $Z_{\frac{1}{d}}$ — and set Φ is a singleton comprehending formula $Z_{\frac{1}{d}} \leftrightarrow \neg(d-1)Z_{\frac{1}{d}}$ — from now on denoted by $\varphi_{\frac{1}{d}}$. In fact, for any valuation $v\in\mathbf{Val}_{\varphi_{\frac{1}{d}}}$, $v(Z_{\frac{1}{d}}) = \frac{1}{d}$. Moreover, we have the following result.

Lemma 1 *Given a rational number $c\in[0,1]$, there is a set Φ of formulas, with $Z_c\in Var(\Phi)$, such that, for any valuation $v\in\mathbf{Val}_\Phi$, we have $v(Z_c) = c$.* □

Proof The result already holds in Ł$_\infty$ for $c = 0$ and $c = 1$; also, it was established for $c = \frac{1}{b}$, with $b\in\mathbb{N}^*$, in previous discussion. Let $c = \frac{a}{b}$, with $a, b\in\mathbb{Z}$ and $0 < a < b$, and $\varphi_c = Z_c \leftrightarrow aZ_{\frac{1}{b}}$. If $\varphi_{\frac{1}{b}}, \varphi_c\in\Phi$, any valuation $v\in\mathbf{Val}_\Phi$ makes $v(Z_c) = c$. ■

Our next step is to show that *truncated linear functions* are representable in Ł$_\infty$-MODSAT. Let

$g : [0,1]^n \to [0,1]$ be a function and $\mathbf{x} \in [0,1]^n$, we write its truncated version by

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

Lemma 2 *Let $g : [0,1]^n \to \mathbb{R}$ be a linear function with rational coefficients,*

$$g(\mathbf{x}) = \frac{a_1}{b_1}x_1 + \cdots + \frac{a_n}{b_n}x_n + c,$$

where $a_i \in \mathbb{Z}$, $b_i \in \mathbb{Z}^*_+$ and $c \in \mathbb{Q}$. Then, $g^{\#}$ is representable in Ł$_\infty$-MODSAT. □

Proof We proceed by induction on $a = |a_1| + \cdots + |a_n|$. If $a = 0$, the result follows by Lemma 1. For $a > 0$, assume the lemma holds for $a - 1$ and, with no loss of generality, that $|a_1| = \max(|a_1|, \ldots, |a_n|)$.

Let us consider first the case where $a_1 > 0$. Let $h = g - \frac{x_1}{b_1}$, such that

$$h(\mathbf{x}) = \frac{a_1 - 1}{b_1}x_1 + \cdots + \frac{a_n}{b_n}x_n + c.$$

By induction hypothesis, there are $\langle \varphi_h, \Phi_h \rangle$ and $\langle \varphi_{h+1}, \Phi_{h+1} \rangle$ which represent $h^{\#}$ and $(h+1)^{\#}$, respectively. We define

$$\Phi = \Phi_h \cup \Phi_{h+1} \cup \Big\{ Z_{\frac{1}{b_1}} \leftrightarrow \neg(b_1 - 1)Z_{\frac{1}{b_1}}, \quad b_1 Z_1 \leftrightarrow X_1, \quad Z_1 \to Z_{\frac{1}{b_1}} \Big\},$$

and claim that $\langle \varphi, \Phi \rangle$, with $\varphi =_{\text{def}} (\varphi_h \oplus Z_1) \odot \varphi_{h+1}$, represents $g^{\#}$. Note that, with the three new formulas added to Φ, the pair $\langle Z_{\frac{1}{b_1}}, \Phi \rangle$ defines the constant $\frac{1}{b_1}$ and the pair $\langle Z_1, \Phi \rangle$ defines function $\frac{x_1}{b_1}$, depending on the value of $x_1 = v(X_1)$; this remark together with the induction hypothesis and by truth-functionality yield that $\mathbf{X}_n$ determines φ modulo Φ-satisfiable. Let $v \in \mathbf{Val}_\Phi$ and $\mathbf{x} = \langle v(X_1), \ldots, v(X_n) \rangle \in [0,1]^n$. When $\mathbf{x}$ is such that $h(\mathbf{x}) \in [0,1]$,

$$g^{\#}(\mathbf{x}) = \Big(h(\mathbf{x}) + \frac{x_1}{b_1}\Big)^{\#} = v(\varphi_h \oplus Z_1) = v((\varphi_h \oplus Z_1) \odot \mathbf{1}) = v(\varphi).$$

When $\mathbf{x}$ is such that $h(\mathbf{x}) \in [-1,0]$,

$$\begin{aligned} g^{\#}(\mathbf{x}) &= \Big(h(\mathbf{x}) + \frac{x_1}{b_1}\Big)^{\#} = \max\Big(0, h(\mathbf{x}) + \frac{x_1}{b_1}\Big) = \max\Big(0, \frac{x_1}{b_1} + h(\mathbf{x}) + 1 - 1\Big) = \\ &= v(Z_1 \odot \varphi_{h+1}) = v(\varphi). \end{aligned}$$

The cases where $h(\mathbf{x}) > 1$ and $h(\mathbf{x}) < -1$ are trivial; then, φ represents $g^{\#}$ modulo Φ-satisfiable.

For the case where $a_1 < 0$, it is sufficient to apply the same reasoning to $1 - g$. As $1 - (1-g)^{\#} = g^{\#}$, the lemma follows. ■

Next, we prove a version of our representation theorem for the simpler case of rational McNaughton functions with one variable; the simplification is intended to better illustrate the role of hat functions.

Theorem 2 *Let $f : [0,1] \to [0,1]$ be a one-variable rational McNaughton function. Then, f is representable in Ł$_\infty$-MODSAT.* □

PROOF The domain $[0,1]$ of f may be partitioned into $[\alpha_i, \alpha_{i+1}]$, $i = 0, \ldots, n-1$, such that each restriction $f|_{[\alpha_i,\alpha_{i+1}]}$ is a linear piece that constitutes f; let $\beta_i = f(\alpha_i)$.

We define the *hat functions* $\mathcal{H}_i : [0,1] \to [0,1]$, $i = 0, \ldots, n$, by:

- $\mathcal{H}_0$ has as graph the segments from $\langle \alpha_0, \beta_0 \rangle$ to $\langle \alpha_1, 0 \rangle$ and from $\langle \alpha_1, 0 \rangle$ to $\langle \alpha_n, 0 \rangle$;
- $\mathcal{H}_i$ has as graph the segments from $\langle \alpha_0, 0 \rangle$ to $\langle \alpha_{i-1}, 0 \rangle$, from $\langle \alpha_{i-1}, 0 \rangle$ to $\langle \alpha_i, \beta_i \rangle$, from $\langle \alpha_i, \beta_i \rangle$ to $\langle \alpha_{i+1}, 0 \rangle$ and from $\langle \alpha_{i+1}, 0 \rangle$ to $\langle \alpha_n, 0 \rangle$, $i = 1, \ldots, n-1$;
- $\mathcal{H}_n$ has as graph the segments from $\langle \alpha_0, 0 \rangle$ to $\langle \alpha_{n-1}, 0 \rangle$, from $\langle \alpha_{n-1}, 0 \rangle$ to $\langle \alpha_n, \beta_n \rangle$.

The hat functions has graphs as in Figure 3.3; such functions are only different from the Schauder hats in Mundici (1994) on the values $\beta_i \in \mathbb{Q}$.

By Lemma 2, hat functions $\mathcal{H}_0$ and $\mathcal{H}_n$ are easily representable in Ł$_\infty$-MODSAT since they may be written as a function $g^{\#}$, where g is linear. The other hat functions $\mathcal{H}_i$, $i = 1, \ldots, n-1$, may be represented by pairs $\langle \varphi_1 \wedge \varphi_2, \Phi_1 \cup \Phi_2 \rangle$, where $\langle \varphi_1, \Phi_1 \rangle$ and $\langle \varphi_2, \Phi_2 \rangle$ represent $g_1^{\#}$ and $g_2^{\#}$, respectively, where g_1 and g_2 are linear. Note that the variables Z_1 associated to the variable x_1, with intention to have value $\frac{x_1}{b_1}$ as in Lemma 2, must be different for each representation $\langle \varphi_1, \Phi_1 \rangle$ and $\langle \varphi_2, \Phi_2 \rangle$.

Let $\langle \varphi_{\mathcal{H}_i}, \Phi_{\mathcal{H}_i} \rangle$ be a representation of $\mathcal{H}_i$. Then, f is representable in Ł$_\infty$-MODSAT by the pair $\langle \varphi_{\mathcal{H}_1} \oplus \cdots \oplus \varphi_{\mathcal{H}_n}, \Phi_{\mathcal{H}_1} \cup \cdots \cup \Phi_{\mathcal{H}_n} \rangle$. The same note about variables Z_1 in the former paragraph also applies here. ■

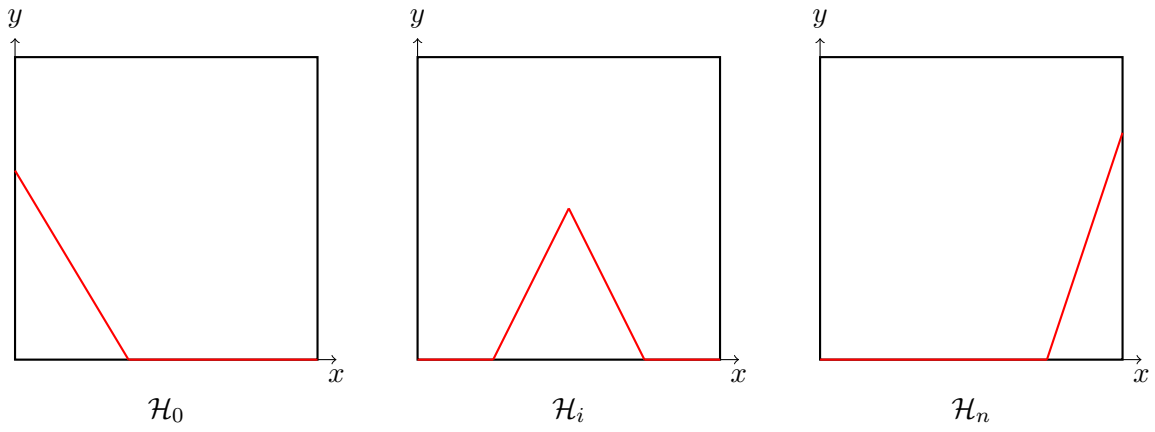


Figure 3.3: *Graphs of examples of one-variable hat functions $\mathcal{H}_0$, $\mathcal{H}_i$, for $i = 1, \ldots, n-1$, and $\mathcal{H}_n$*

In the following, we prove the main result of this section which generalizes Theorem 2 to the multivariate case; its proof uses constructions from the literature (see Section 2.4 for references) and subsume the use of hat functions above.

Theorem 3 *Let $f : [0,1]^n \to [0,1]$ be a (multivariable) rational McNaughton function. Then, f is representable in Ł$_\infty$-MODSAT.* □

PROOF The domain of f may be decomposed as follows (Cignoli *et al.*, 2000, Section 3.3). Let $p_1, \ldots, p_m$ be an exhaustive list of distinct linear pieces constituents of f, each pair p_i and p_k of these constituents defines two closed half-spaces H^+ and H^- such that $p_i(\mathbf{x}) \geq p_k(\mathbf{x})$ for $\mathbf{x} \in H^+$

and $p_k(\mathbf{x}) \geq p_i(\mathbf{x})$ for $\mathbf{x} \in H^-$. Thus, for any permutation ρ of the set $\{1, \ldots, m\}$, we define

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

which is a closed convex polyhedron, since it is an intersection of $[0,1]^n$ and a finite set of closed half-spaces. As the p_i's have rational coefficients, the vertices of P_ρ have rational coordinates. Let $\mathcal{W}$ be the set of simplices (also with rational coordinates) arising from some triangulation of n-dimensional polyhedra P_ρ; the union of $\mathcal{W}$ is the cube $[0,1]^n$, the intersection of a pair of elements in $\mathcal{W}$ is either a common face between them or empty and, for each $S \in \mathcal{W}$, there is an index $u_S \in \{1, \ldots, m\}$ such that, restricted to S, $f|_S = p_{u_S}$.

For each vertex $\mathbf{v}$ of some simplex in $\mathcal{W}$, we define the hat function $\mathcal{H}_\mathbf{v} : [0,1]^n \to [0,1]$ so that:

- $\mathcal{H}_\mathbf{v}(\mathbf{v}) = f(\mathbf{v})$;
- $\mathcal{H}_\mathbf{v}(\mathbf{u}) = 0$ for each vertex $\mathbf{u}$ of a simplex in $\mathcal{W}$ different from $\mathbf{v}$;
- $\mathcal{H}_\mathbf{v}$ is linear over each simplex in $\mathcal{W}$.

As in the one-variable case, the hat functions may be represented in Ł$_\infty$-MODSAT by a pair $\langle \varphi, \Phi \rangle$ where φ is a $(\bigvee \bigwedge)$-combination of the hat function linear pieces given by Lemma 2 (Cignoli *et al.*, 2000, Proposition 9.1.4). Thus, f may be represented by $\langle \bigoplus_\mathbf{v} \varphi_{\mathcal{H}_\mathbf{v}}, \bigcup_\mathbf{v} \Phi_{\mathcal{H}_\mathbf{v}} \rangle$. ■

The representations built in Theorems 2 and 3 are said to be in disjunctive normal form since they are disjunctions ($\oplus$) of hat functions, which empowers the modulo satisfiability technique to increase the expressivity of Ł$_\infty$. Moreover, since they are constructive proofs, we might be tempted to derive algorithms for building representations in Ł$_\infty$-MODSAT from them. However, such representations could be unnecessarily complex for the following reasons.

- For m distinct linear pieces, there may be $m!$ n-dimensional polyhedra P_ρ in the worst case.
- The number of simplices in a minimum-cardinality decomposition of the n-dimensional unit cube may be too high; for instance, for $[0,1]^7$ it is at least 1175 (Hughes and Anderson, 1996).
- Besides that, the representations of truncated linear functions in Lemma 2 are already exponential in the binary representation of their coefficients, since they are inductively built on $a = |a_1| + \cdots + |a_n|$.

3.4 An Efficient Algorithm for Building Representations

We have showed that the coupling of representation modulo satisfiability with hat functions enables the constructive representation of rational McNaughton functions in Ł$_\infty$-MODSAT. However, we need to produce a less complex representation in order to derive an efficient algorithm that actually builds it; this is our aim in this section.

3.4.1 Regional Format of Rational McNaughton Functions

In representation via hat functions, a rational McNaughton function is first seen as a partition of its domain in subsets P_ρ in a way that, in each subset, the function is identical to one of its

linear pieces. For our algorithm, we standardize an encoding of rational McNaughton functions that encompasses this format, however is more general.

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

$$p_i(\mathbf{x}) = \gamma_{i0} + \gamma_{i1}x_1 + \cdots + \gamma_{in}x_n, \tag{3.2}$$

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 $\Omega_i \subseteq [0,1]^n$ called *region* such that:

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

Note that in the regional format encoding we allow the repetition of linear pieces so that there is a one-to-one correspondence between regions and them. 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 assumed encoding. We discuss the regional format further at the end of this section.

Example 4 Rational McNaughton function f with graph in Figure 3.4 may be given by the linear pieces:

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

Regions Ω_i associated to each linear piece are depicted in Figure 3.5a and described in Table 3.1; we soon tackle the problem of deciding if a linear piece is above another. The polyhedra P_ρ in the representation of f via hat functions (Section 3.3) are depicted in Figure 3.5b; note that, for such representation, these polyhedra still need a further decomposition into triangles (2-simplices). □

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

Lemma 3 *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 $E \subseteq \partial U$ and, since U is a finite

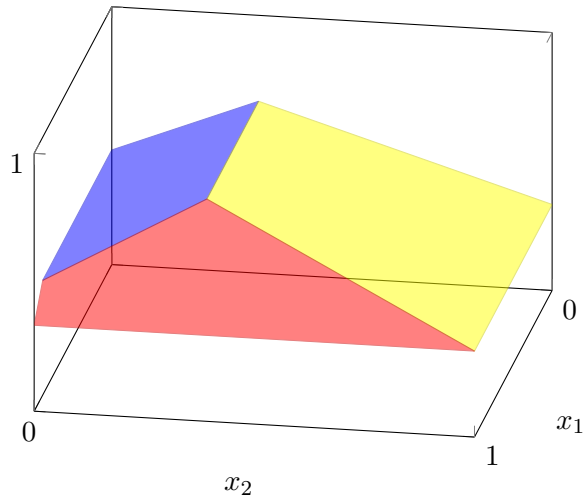
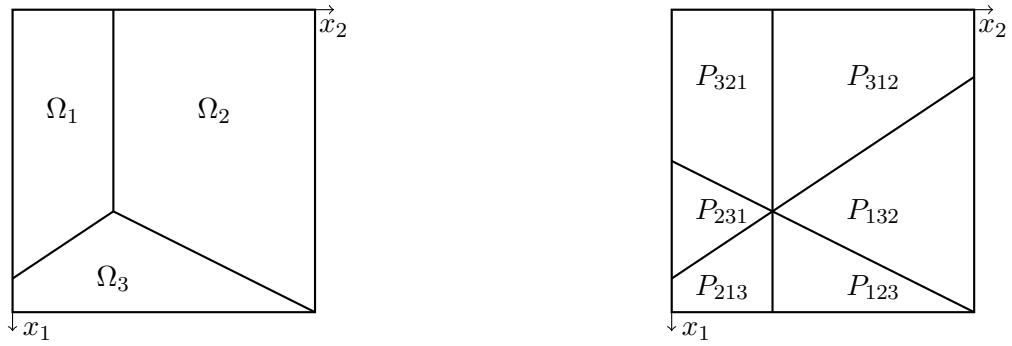


Figure 3.4: *Graph of rational McNaughton function with three linear pieces over* $[0,1]^2$



(a) *Config. with* 3 *regions*

(b) *Config. with* 6 *regions*

Figure 3.5: *Some possible region configurations for function* f *in Example 4*

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

Table 3.1: *Regions* Ω_i *for function* f *in Example 4*

union of regions Ω_i, there exists an infinite set $E' \subseteq E$, such that $E' \subseteq \partial\Omega' \subseteq \mathrm{cl}(\Omega')$, for some $\Omega' = \Omega_i \neq \Omega$. Let $E_{n+1} \subseteq E'$ be a set with $n+1$ points; as $\mathrm{cl}(\Omega)$ and $\mathrm{cl}(\Omega')$ are convex sets, we have that $\mathrm{conv}(E_{n+1}) \subseteq \mathrm{cl}(\Omega) \cap \mathrm{cl}(\Omega')$. 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 $\Omega^\circ \cap \Omega'^\circ \neq \varnothing$, 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

$$\mathrm{cl}(\Omega) = \Big\{ \mathbf{x} \in [0,1]^n \;\Big|\; \omega_{j0} + \omega_{j1}x_1 + \cdots + \omega_{jn}x_n \geq 0,\ j = 1, \ldots, \lambda_\Omega \Big\}. \tag{3.3}$$

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 (3.3) as input. Let p_k and p_i be given by

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

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:

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

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 (see Section 2.4 for references).

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.

```
1: M := MAX(p_i − p_k, cl(Ω_i));
2: if M ≤ 0 then
3:         return True;
4: else
5:         return False;
6: end if
```

Theorem 4 *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,

$$p_i(\mathbf{x}) - p_k(\mathbf{x}) \leq 0, \tag{3.4}$$

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 (3.4)

is satisfied by all $\mathbf{x} \in \Omega_i \subseteq \mathrm{cl}(\Omega_i)$. In case $M > 0$, then, either $\mathbf{x}_M \in \Omega_i$ fails to fulfill (3.4) 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 4, it is enough to encode regions Ω in such a way that there is a polynomial procedure to compute $\mathrm{cl}(\Omega)$ as in (3.3). 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 (3.3). 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 $p_1, \ldots, p_{\bar{m}}$. As in representation via hat functions in Section 3.3, for each permutation ρ of the set $\{1, \ldots, \bar{m}\}$, we define the polyhedron

$$P_\rho = \Big\{ \mathbf{x} \in [0,1]^n \;\Big|\; p_{\rho(1)}(\mathbf{x}) \geq \cdots \geq p_{\rho(\bar{m})}(\mathbf{x}) \Big\}. \tag{3.5}$$

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

Theorem 5 *The set $\mathcal{C}$ has the following properties.*

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

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

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

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

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

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

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

(d) Let $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, \bar{m}\}$ be a non-singleton set of indexes such that for any $l \in \{1, \ldots, k\}$, there is a $\mathbf{x} \in P^\circ$, such that $f(\mathbf{x}) = p_{i_l}(\mathbf{x})$. Let $U_{i_l} = \{\mathbf{x} \in P^\circ \mid f(\mathbf{x}) = p_{i_l}(\mathbf{x})\} \neq \varnothing$, for $l = 1, \ldots, k$; we have that $\cup_{l=1}^{k} U_{i_l} = P^\circ$ and, by item (b), $U_{i_{l'}} \cap U_{i_{l''}} = \varnothing$, for $l' \neq l''$. As

P° is a connected set, there are distinct $i', i'' \in \{i_1, \ldots, i_k\}$ and $\mathbf{b} \in P^\circ$ such that $\mathbf{b} \in \partial U_{i'}$ and $\mathbf{b} \in U_{i''}$. As $p_{i'}$ restricted to $U_{i'} \cup \{\mathbf{b}\}$ is continuous, for any sequence $\{\mathbf{b}_n\} \subseteq U_{i'}$ such that $\lim \mathbf{b}_n = \mathbf{b}$ (which exists since $\mathbf{b} \in \partial U_{i'}$), we have that $\lim f(\mathbf{b}_n) = \lim p_{i'}(\mathbf{b}_n) = p_{i'}(\mathbf{b})$. However, $f(\mathbf{b}) = p_{i''}(\mathbf{b}) \neq p_{i'}(\mathbf{b})$, by item (b), contrary to the continuity of f. Therefore, there is only one $i_P \in \{1, \ldots, \bar{m}\}$ such that $f(\mathbf{x}) = p_{i_P}(\mathbf{x})$, for $\mathbf{x} \in P^\circ$ and, by continuity of f, for all $\mathbf{x} \in P$.

(e) 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 $[\mathbf{d}', \mathbf{d}] \subseteq [\mathbf{a}, \mathbf{d}]$; 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 $\mathcal{C}$ 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 associated to many distinct polyhedra. Determining whether a linear piece p_k is above other linear piece p_i over $P \in \mathcal{C}$ 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 3.5b shows the permutation-based configuration $\mathcal{C}$ 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 we adapted in Section 3.3 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 $\mathcal{C}$; this path has also been followed in the literature for representing continuous piecewise linear functions in other Ł$_\infty$-based logical systems; see Section 3.7.

As already noticed, the setback with describing a rational McNaughton function using the set $\mathcal{C}$ of polyhedra is that, in the worst case, $|\mathcal{C}| = \bar{m}!$; 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 3.5 shows such contrast related to Example 4 and it may also be seen in the classes of functions in Section 3.5. This feature does not interfere with the complexity of the general algorithm in Section 3.4.3, 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 make some applications viable. Of course, 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.

3.4.2 A Particular Case: Truncated Linear Functions

In Lemma 2, we have a constructive representation of truncated linear functions — which are particular cases of rational McNaughton functions — in Ł$_\infty$-MODSAT; unfortunately, this is an exponential construction as it is based on a unary representation of a_i. We show next another possibility for representing these functions and provide a polynomial algorithm for computing them.

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

$$p(\mathbf{x}) = \frac{a_0}{b_0} + \frac{a_1}{b_1}x_1 + \cdots + \frac{a_n}{b_n}x_n, \tag{3.6}$$

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

$$p^\#(\mathbf{x}) = \min\Big(1, \max\big(0, p(\mathbf{x})\big)\Big). \tag{3.7}$$

We have that $p^\#(\mathbf{x}) = 0$, if $p(\mathbf{x}) < 0$; $p^\#(\mathbf{x}) = 1$, if $p(\mathbf{x}) > 1$; and $p^\#(\mathbf{x}) = p(\mathbf{x})$, otherwise. In order to rewrite expression (3.6), we define:

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

where $j \in P$, if $a_j > 0$, and $j \in N$, if $a_j < 0$, with $P \cup N \subseteq \{0, \ldots, n\}$, 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:

$$p_P(\mathbf{x}) = \sum_{j\in P}\frac{\alpha_j}{\beta_j}x_j; \qquad\qquad p_N(\mathbf{x}) = \sum_{j\in N}\frac{\alpha_j}{\beta_j}x_j. \tag{3.8}$$

Lemma 4 *Functions p, p_P and p_N in (3.6) and (3.8) have the following properties, for $\mathbf{x} \in [0,1]^n$:*

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

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

Proof By elementary algebraic manipulation. ■

The lemma above decomposes function p in terms of p_P and p_N. Let us represent the latter ones. Let $Z_j^p, Z_{\frac{1}{\beta_j}} \in \mathbb{P}$; for a set of indexes $J \in \{P, N\}$, define:

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

And then, define:

$$\begin{array}{lll} \bar{\varphi}_J = \tilde{\varphi}_J, & \bar{\Phi}_J = \tilde{\Phi}_J, & \text{if } 0 \notin J; \\ \bar{\varphi}_J = \alpha_0 Z_{\frac{1}{\beta_0}} \oplus \tilde{\varphi}_J, & \bar{\Phi}_J = \tilde{\Phi}_J \cup \{\varphi_{\frac{1}{\beta_0}}\}, & \text{otherwise.} \end{array} \tag{3.9}$$

Lemma 5 *Functions p_P and p_N in (3.8) may respectively be represented by $\langle \bar{\varphi}_P, \bar{\Phi}_P \rangle$ and $\langle \bar{\varphi}_N, \bar{\Phi}_N \rangle$ in (3.9).* □

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

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

by Lemma 4 and aforementioned equations $v(Z_{\frac{1}{\beta_0}}) = \frac{1}{\beta_0}$ and $\beta_j \cdot v(Z_j^p) = v(X_j)$. The case where $0 \notin J$ is similar. ■

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

$$\bar{\varphi}_p = \beta[\neg(\bar{\varphi}_P \rightarrow \bar{\varphi}_N)], \qquad \bar{\Phi}_p = \bar{\Phi}_P \cup \bar{\Phi}_N. \tag{3.10}$$

Theorem 6 *Function $p^{\#}$ in (3.7) may be represented by $\langle \bar{\varphi}_p, \bar{\Phi}_p \rangle$ in (3.10).* □

Proof For $\langle x_1, \ldots, x_n \rangle \in [0,1]^n$, there exists $v \in \mathbf{Val}_{\bar{\Phi}_p}$ such that $v(X_j) = x_j$ as in the proof of Lemma 5 with $J = P \cup N$. Now, let $v, v' \in \mathbf{Val}_{\bar{\Phi}_p}$ such that $v(X_j) = v'(X_j)$, for $j = 1, \ldots, n$. In particular, $v, v' \in \mathbf{Val}_{\bar{\Phi}_J}$ and, by Lemma 5, $v(\bar{\varphi}_J) = v'(\bar{\varphi}_J)$, for $J \in \{P, N\}$. Therefore, $v(\bar{\varphi}_p) = v'(\bar{\varphi}_p)$ and $\mathbf{X}_n$ determines $\bar{\varphi}_p$ modulo $\bar{\Phi}_p$-satisfiable. Finally, suppose $v \in \mathbf{Val}_{\bar{\Phi}_p}$. In particular, $v \in \mathbf{Val}_{\bar{\Phi}_P}$ and $v \in \mathbf{Val}_{\bar{\Phi}_N}$. If $p(v(X_1), \ldots, v(X_n)) \leq 0$, by Lemma 4, $p_P(v(X_1), \ldots, v(X_n)) \leq p_N(v(X_1), \ldots, v(X_n))$. Therefore, by Lemma 5, $v(\bar{\varphi}_P) \leq v(\bar{\varphi}_N)$ and, then, $v(\bar{\varphi}_p) = 0$. On the other hand, if $p(v(X_1), \ldots, v(X_n)) \geq 0$, by Lemma 4, $p_P(v(X_1), \ldots, v(X_n)) \geq p_N(v(X_1), \ldots, v(X_n))$. Therefore, by Lemma 5, $v(\bar{\varphi}_P) \geq v(\bar{\varphi}_N)$ and, then, $v(\neg(\bar{\varphi}_P \rightarrow \bar{\varphi}_N)) = 1 - \min(1, 1 - v(\bar{\varphi}_P) + v(\bar{\varphi}_N)) = v(\bar{\varphi}_P) - v(\bar{\varphi}_N)$. Finally, by Lemmas 4 and 5, $p(v(X_1), \ldots, v(X_n)) = \beta \cdot (v(\bar{\varphi}_P) - v(\bar{\varphi}_N))$, hence $p^{\#}(v(X_1), \ldots, v(X_n)) = v(\bar{\varphi}_p)$ in any case. ■

Table 3.2 shows how functions in Example 4 can be represented as in Theorem 6.

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 $\xi_\psi^0, \xi_\psi^1, \ldots, \xi_\psi^{\lfloor \log n \rfloor}$

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

Table 3.2: *Representations as in (3.10) for functions $p_1^\#$, $p_2^\#$ and $p_3^\#$, where functions p_1, p_2 and p_3 are from Example 4*

and replacing every occurrence of $n\psi$, where $n \in \mathbb{N} \setminus \{0,1\}$, with the formula

$$\xi_{n\psi} =_{\text{def}} \bigoplus_{\substack{k=0 \\ n_k=1}}^{\lfloor \log n \rfloor} \xi_\psi^k, \tag{3.11}$$

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

$$\begin{array}{ll} \xi_\psi^0 & \leftrightarrow \psi; \\ \xi_\psi^k & \leftrightarrow \xi_\psi^{k-1} \oplus \xi_\psi^{k-1}, \text{ for } k = 1, \ldots, \lfloor \log n \rfloor. \end{array} \tag{3.12}$$

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 7.

Lemma 6 *Let $n \in \mathbb{N} \setminus \{0,1\}$, ψ be a formula and $\xi_{n\psi}$ and $\Xi_{n\psi}$ be respectively a formula as in (3.11) and a set as in (3.12) built from n and ψ. For any valuation $v \in \mathbf{Val}_{\Xi_{n\psi}}$, $v(n\psi) = v(\xi_{n\psi})$.* □

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

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

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

Theorem 7 *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 (3.10) by replacing any occurrence of $n\psi$ in $\bar{\varphi}_p$ and $\bar{\Phi}_p$ with $\xi_{n\psi}$ in (3.11) and by adding formulas in set $\Xi_{n\psi}$ in (3.12) to $\bar{\Phi}_p$. Then, $\langle \varphi_p, \Phi_p \rangle$ is also a representation for $p^\#$ in (3.7). 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 For $\langle x_1, \ldots, x_n \rangle \in [0,1]^n$, define a valuation v such that $v(X_j) = x_j$ and $v(Z_j^p) = \frac{x_j}{\beta_j}$, for $j = 1, \ldots, n$, $v(Z_{\frac{1}{\beta_j}}) = \frac{1}{\beta_j}$, for $j = 0, \ldots, n$, $v(\xi_\psi^0) = v(\psi)$ and $v(\xi_\psi^k) = \min(1, v(\xi_\psi^{k-1}) + v(\xi_\psi^{k-1}))$, for $k = 1, \ldots, \lfloor \log n \rfloor$. Note that $v \in \mathbf{Val}_{\bar{\Phi}_p}$ and $v \in \mathbf{Val}_{\Xi_{n\psi}}$, then, by Lemma 6, as $\Xi_{n\psi} \subseteq \Phi_p$, we have that $v \in \mathbf{Val}_{\Phi_p}$. Still, for any $v \in \Phi_p$, we have that $v \in \mathbf{Val}_{\Xi_{n\psi}}$ and, by Lemma 6, $v \in \bar{\Phi}_p$. Therefore, again by Lemma 6, for $v, v' \in \Phi_p$ such that $v(X_j) = v'(X_j)$, for $j = 1, \ldots, n$, it follows that $v(\varphi_p) = v'(\varphi_p)$, $\mathbf{X}_n$ determines φ_p modulo Φ_p-satisfiable and $p^\#(v(X_1), \ldots, v(X_n)) = v(\varphi_p)$. This argument still holds when considering multiple replacements. ■

We set $\langle \varphi_p, \Phi_p \rangle$ from $\langle \bar{\varphi}_p, \bar{\Phi}_p \rangle$ in (3.10) by properly replacing all occurrences of $n_l\psi_l$ as stated in the above theorem. By construction, $\langle \varphi_p, \Phi_p \rangle$ is given by

$$\varphi_p = \beta[\neg(\varphi_P \to \varphi_N)]; \qquad \Phi_p = \Phi_P \cup \Phi_N; \tag{3.13}$$

where φ_P, φ_N, Φ_P and Φ_N are properly defined from their barred correspondents in (3.9). Table 3.3 shows how functions in Example 4 can be represented as in Theorem 7.

Algorithm 2 BINARY-F: computes formula $\xi_{n\psi}$ in (3.11) or $\mathbf{0}$ or ψ

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

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

Algorithm 3 BINARY-S: computes set $\Xi_{n\psi}$ in (3.12) or $\varnothing$

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

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

Algorithms 2 and 3 compute the representation of $n\psi$ in Ł$_\infty$-MODSAT. Algorithm 2 returns $\mathbf{0}$ and ψ in the limit cases $n = 0$ and $n = 1$ (lines 1 to 5); when $n \in \mathbb{N} \setminus \{0, 1\}$, it returns formula $\xi_{n\psi}$ in (3.11) by building it in line 6 plus a $\lfloor \log n \rfloor + 1$ iteration loop (lines 7 to 13) where the n_k's in the binary representation of n are calculated by the routine in lines 8 and 9. Algorithm 3 returns $\varnothing$ in the limit cases $n = 0$ and $n = 1$ (lines 1 to 3); when $n \in \mathbb{N} \setminus \{0, 1\}$, it returns set $\Xi_{n\psi}$ that comprehends formulas (3.12) by building it in line 4 plus a $\lfloor \log n \rfloor$ iteration loop (lines 5 to 7). Both algorithms terminate in time $O(\log n)$ assuming propositional variables are all represented with a constant size.

Algorithm 4 computes a representation of $p^\#$ in Ł$_\infty$-MODSAT. It returns $\langle \mathbf{0}, \varnothing \rangle$ in the limit case $a_0 = \cdots = a_n = 0$ (lines 1 to 3); otherwise it returns representation $\langle \varphi_p, \Phi_p \rangle$ given in (3.13). From line 4 to line 15, the algorithm sets all P, N, α_j, β_j and β, for $j = 0, \ldots, n$, which are used to rewrite function p in terms of p_P and p_N as in Lemma 4. From line 16 to line 26, it writes formulas φ_P and φ_N and adds formulas in Φ_P and Φ_N to Φ_p. For $J \in \{P, N\}$, it works throughout a $|J|$ iteration loop where each iteration takes a coefficient $\frac{a_j}{b_j}$ into account, where it treats $\frac{a_0}{b_0}$ (lines 18

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

Table 3.3: *Representations as in (3.13) for functions $p_1^\#$, $p_2^\#$ and $p_3^\#$, where functions p_1, p_2 and p_3 are from Example 4*

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

to 21) separately from the others (lines 22 to 25). In lines 27 and 28 it finally writes formula φ_p and completes set Φ_p.

Theorem 8 *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 Algorithm 4 builds representation $\langle \varphi_p, \Phi_p \rangle$ in (3.13). So, its correctness follows from Theorem 7. Let $[0,1]^n$ be the domain of p and M the maximum size of a binary representation for numbers among a_j and b_j; then the input size of p is at most $2(n+1)M$. The algorithm first calculates in polynomial time all β, α_j and β_j; let μ be the maximum size of a binary representation for numbers among β, α_j and β_j. Then, it proceeds to writing the representation which is made up of at most $3(n+1)$ propositional variables of the type X_j, Z_j^p and $Z_{\frac{1}{\beta_j}}$ and $2(n+1)\mu+\mu$ propositional variables of the type ξ_ψ^k, a quantity polynomially proportional to the size of the input; thus, the size of the representation for each propositional variable may be assumed to be a constant π also polynomially proportional to the size of the input. The algorithm calculates formulas φ_P and φ_N and sets Φ_P and Φ_N in $n+1$ steps; in each one it calculates the part associated to a coefficient $\frac{\alpha_i}{\beta_i}$. For each part, computation takes polynomial time on π and at most six executions of routines BINARY-F (Algorithm 3) and BINARY-S (Algorithm 2) with argument $\langle \nu, P \rangle$, where ν is α_i, β_i or $\beta_i - 1$, which are already or may be quickly computed, and P is a propositional variable. In these cases BINARY-F and BINARY-S run in polynomial time on μ and π. The algorithm finishes calculating φ_p and Φ_p by running BINARY-F and BINARY-S with argument $\langle \beta, \neg(\varphi_P \to \varphi_N) \rangle$. Now, BINARY-F runs in polynomial time on μ and π and BINARY-S runs in polynomial time on μ, π and the size of $\neg(\varphi_P \to \varphi_N)$. After all, Algorithm 5 terminates in polynomial time. ■

We call REPRESENT-TL-F and REPRESENT-TL-S the routines that separately compute φ_p and Φ_p, respectively. Both may be easily derived from routine REPRESENT-TL in Algorithm 4.

3.4.3 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:

$$p_i(\mathbf{x}) = \frac{a_{i0}}{b_{i0}} + \frac{a_{i1}}{b_{i1}} x_1 + \cdots + \frac{a_{in}}{b_{in}} x_n, \tag{3.14}$$

for $\mathbf{x} = \langle x_1, \ldots, x_n \rangle \in [0,1]^n$, $a_{ij} \in \mathbb{Z}$, $b_{ij} \in \mathbb{Z}_+^*$ and $i = 1, \ldots, m$, with each piece identical to f in region Ω_i, for $i = 1, \ldots, m$. We call ABOVE(p_k,p_i) the polynomial 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 7, for $i = 1, \ldots, m$. We define

$$\varphi = \bigvee_{i=1}^{m} \varphi_{\Omega_i}, \text{ with } \varphi_{\Omega_i} = \bigwedge_{k \in K_{\Omega_i}} \varphi_{p_k}; \qquad \Phi = \bigcup_{i=1}^{m} \Phi_{p_i}; \tag{3.15}$$

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

Lemma 7 *Let f be a rational McNaughton function in regional format with linear pieces given by (3.14) and let φ_{Ω_j} be a formula and Φ a set as in (3.15). Then, $v(\varphi_{\Omega_j}) \leq f(v(X_1), \ldots, v(X_n))$, for $v \in \mathbf{Val}_\Phi$.* □

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

Table 3.4: *Representation as in* (3.15) *for function f from Example 4*

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 7,

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

If $\mathbf{x}_0 \in \Omega_j$, then $v(\varphi_{\Omega_j}) \leq p_j^{\#}(\mathbf{x}_0) = p_j(\mathbf{x}_0) = f(\mathbf{x}_0)$. On the 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 9 *Any rational McNaughton function may be represented by $\langle \varphi, \Phi\rangle$ in* (3.15). □

Proof First note that any rational McNaughton function may be put in regional format as showed in Section 3.4.1. 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 $v(Z_j^{p_i}) = \frac{x_j}{\beta_{ij}}$, for $i = 1, \ldots, m, j = 1, \ldots, n$, $v(Z_{\frac{1}{\beta_{ij}}}) = \frac{1}{\beta_{ij}}$, for $i = 1, \ldots, m, j = 0, \ldots, n$, $v(\xi_\psi^0) = v(\psi)$ and $v(\xi_\psi^k) = \min(1, v(\xi_\psi^{k-1}) + v(\xi_\psi^{k-1}))$, for $k = 1, \ldots, \lfloor \log n \rfloor$, for any $n\psi$ that occurs in φ and Φ. Now, let $v, v' \in \mathbf{Val}_\Phi$ such that $v(X_j) = v'(X_j)$, for $j = 1, \ldots, n$. In particular, $v, v' \in \mathbf{Val}_{\Phi_{p_i}}$, for $i = 1, \ldots, m$, and, by Theorem 7, $v(\varphi_{p_i}) = v'(\varphi_{p_i})$, 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 7. ■

Table 3.4 shows how function f in Example 4 can be represented as in Theorem 9.

Algorithm 5 returns representation $\langle \varphi, \Phi\rangle$ for function f with linear pieces given in (3.14). From line 1 to line 13, the algorithm writes formulas φ_{Ω_i} and the set Φ: it first computes formulas φ_{p_i} (lines 2 to 5) by means of routine REPRESENT-TL-F and then it writes φ_{Ω_i} (lines 7 to 11) by means of routine ABOVE. It writes set Φ computing each Φ_{p_i} by means of routine REPRESENT-TL-S (line 12). In line 14 it writes formula φ.

Theorem 10 *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 Algorithm 5 builds representation $\langle \varphi, \Phi\rangle$ in (3.15). So, the algorithm correctness follows from Theorem 9. The size of f is the space necessary to storage the coefficients of its m linear pieces $p_1, \ldots, p_m$ and the regions $\Omega_1, \ldots, \Omega_m$. The algorithm first calculates m representative formulas φ_{p_i} by REPRESENT-TL-F, which takes polynomial time on the size of f by Theorem 8. Then, it

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: $\Phi := \varnothing$;
2: **for** $i = 1, \ldots, m$ **do**
3: $\quad \varphi_{p_i} :=$ REPRESENT-TL-F$(\frac{a_{i0}}{b_{i0}}, \ldots, \frac{a_{in}}{b_{in}})$;
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$(\frac{a_{i0}}{b_{i0}}, \ldots, \frac{a_{in}}{b_{in}})$;
13: **end for**
14: $\varphi := \varphi_{\Omega_1} \vee \cdots \vee \varphi_{\Omega_m}$;
15: **return** $\langle \varphi, \Phi \rangle$;

builds formulas φ_{Ω_i} from the already built representative formulas in m^2 steps; in each of these steps it runs routine ABOVE in assumed polynomial time. Along with the above computation, the algorithm also builds set Φ in m steps; in each one it calculates set Φ_{p_i} by REPRESENT-TL-S, which takes polynomial time on the size of f by Theorem 8. Finally, the algorithm calculates φ from formulas φ_{Ω_i} already computed. After all, Algorithm 5 terminates in polynomial time. ■

Summarizing, while the representation of a rational McNaughton function f via hat functions has a disjunction of hat functions in its representative formula $\varphi_{\oplus}$, the representation just presented has a $(\bigvee\bigwedge)$-combination of linear pieces in its representative formula $\varphi_{\vee\wedge}$.

Let f with $\bar{m}$ distinct linear pieces; on the one hand, $\varphi_{\oplus}$ is a disjunction of as many hat functions as vertices in the decompositions in simplices of the polyhedra in $\mathcal{C}$; regarding that the size of set $\mathcal{C}$ is $\bar{m}!$ in the worst case, this representation might be highly complex. On the other hand, for an encoding of f in regional format with $m \geq \bar{m}$ regions and also m associated linear pieces — that are allowed to repeat in such encoding —, $\varphi_{\vee\wedge}$ is a $\bigvee$-combination of $\bigwedge$-combinations of the m linear pieces; so there may be at most m^2 occurrences of representative formulas φ_{p_i} of linear pieces. Despite of the possibility of having $m = \bar{m}!$, we have already remarked that, for many cases, it is possible to find a region configuration for which m is significantly smaller than $\bar{m}!$; again, as in Example 4 and in the classes of functions in Section 3.5.

It is worth mentioning that the representation of hat functions in Section 3.3 was also done by means of a $(\bigvee\bigwedge)$-combination of linear functions representative formulas. However, in that context, linear functions are represented in Ł$_\infty$-MODSAT by the exponential representation in Lemma 2; replacing this representation by the one in Section 3.4.2 makes representations of hat functions smaller.

3.4.4 Pre-Regional Format and a Literature Review

The presented algorithm 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 is grounded on Lemma 7 and Theorem 9, 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 the lattice property in such encoding in order to the built representation to be correct.

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 3.4.1, 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 3.6a may be given in pre-regional format by the linear pieces:

- $p_1(x_1, x_2) = p_4(x_1, x_2) = x_2$;
- $p_2(x_1, x_2) = 1 - x_1$;
- $p_3(x_1, x_2) = x_1$;
- $p_5(x_1, x_2) = \frac{1}{4} + \frac{1}{2}x_2$.

Regions Ω_i associated to each linear piece are depicted in Figure 3.6b and described in Table 3.5. 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 3.6b.

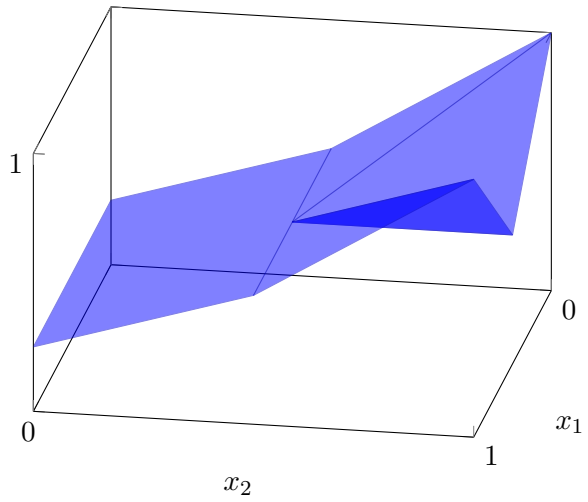
Note that such encoding of f_{CE} does not have the lattice property since there is no linear piece p_k such that p_3 is above p_k over Ω_3 and p_k is above p_5 in Ω_5. Let $\langle \varphi, \Phi \rangle$ be a pair as in (3.15) with intention to be a representation for f_{CE} and let $\mathbf{x}_0 = \langle 0.6, 0.9 \rangle \in \Omega_3^\circ$; we have $K_{\Omega_3} = \{1, 3, 4\}$ 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

$$f_{CE}(\mathbf{x}_0) = p_3(\mathbf{x}_0) = 0.6 < 0.7 = v(\varphi_{\Omega_5}) \leq v(\varphi).$$

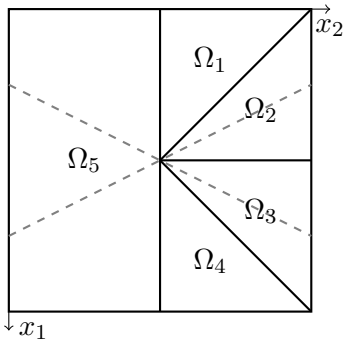
Therefore, $\langle \varphi, \Phi \rangle$ cannot be a representation for function f_{CE} and the lattice property cannot be dropped from regional format in order to perform such representation.

Function f_{CE} may be put in regional format by taking as regions the polyhedra $P_\rho \in \mathcal{C}$ in (3.5); in this case, we have a representation with $|\mathcal{C}| = 9$ 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. □

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 7 and Theorem 9, 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.



(a) *Graph*



(b) *Region configuration*

Figure 3.6: *Graph and region configuration of function f_{CE} in Example 5*

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

Table 3.5: *Regions Ω_i for function f in Example 5*

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 11 *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 $[a,b] \subseteq [0,1]$. 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 $P_\sigma, P_\varsigma \in \mathcal{C}$ be polyhedra as in (3.5) such that there are $\alpha, \beta \in [0,1]$ in a way that $[\alpha, b_i] \subseteq P_\sigma$, $[a_j, \beta] \subseteq P_\varsigma$ and $(\alpha, b_i) \neq \varnothing \neq (a_j, \beta)$. 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

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

Table 3.6: *Number of tests by class of rational McNaughton functions*

the lattice property and it is given in regional format. The result is analogous for the case where $b_j \leq a_i$. ■

3.5 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 (3.7) from a given linear function as in (3.6). 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 (3.6) which are identical to f in given polyhedral regions as in (3.3). 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).

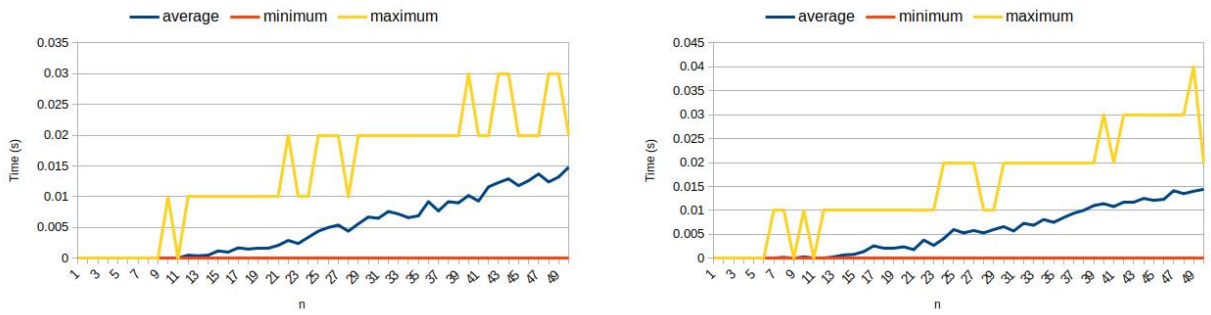
We ran the implementation through experiments in order to measure its execution time and to give evidence for its correctness. The totality of a finite set of tests does not prove correctness, however, in large amounts, it may provide some evidence in favor of it.

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_∞-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).

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 3.6 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.[1]

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



(a) *Running on truncated linear functions*

(b) *Running on normalized linear functions*

Figure 3.7: *Representation builder performance, randomly gen. instances:* $n = 1$ *to* $n = 50$

3.5.1 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 12 *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* $R_1 \subseteq \mathbb{R}^n$ *and* $R_2 \subseteq \mathbb{R}^n$*, 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 (3.7) defined from a linear function p in (3.6). 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 3.7a, 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 (3.6) by the following normalization process performed over $D = [0,1]^n$ by

$$p'(\mathbf{x}) = \frac{p(\mathbf{x}) + \frac{A}{b_0}}{B}, \tag{3.16}$$

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 3.7b, 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 3.8a 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 (3.6) by the normalization process in (3.16) performed over $D = \Omega_1$.

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

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

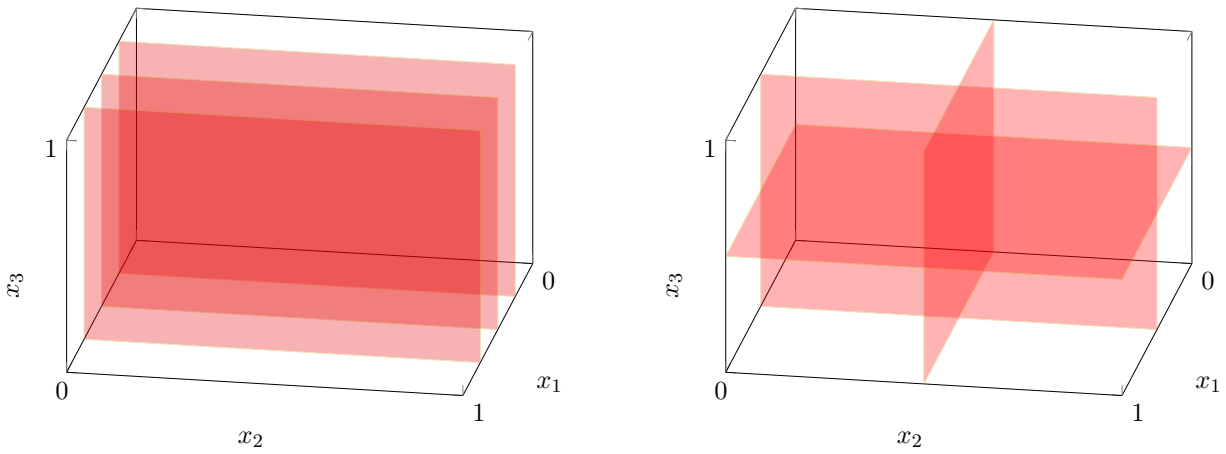
with $q_i \in [-m_i \cdot r, (1 - M_i) \cdot r]$, for

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

These linear pieces and, therefore, function f have range in $[0,1]$; also, function f is continuous over $[0,1]^n$. Theorem 13 below states that such encoding of function f has the lattice property.

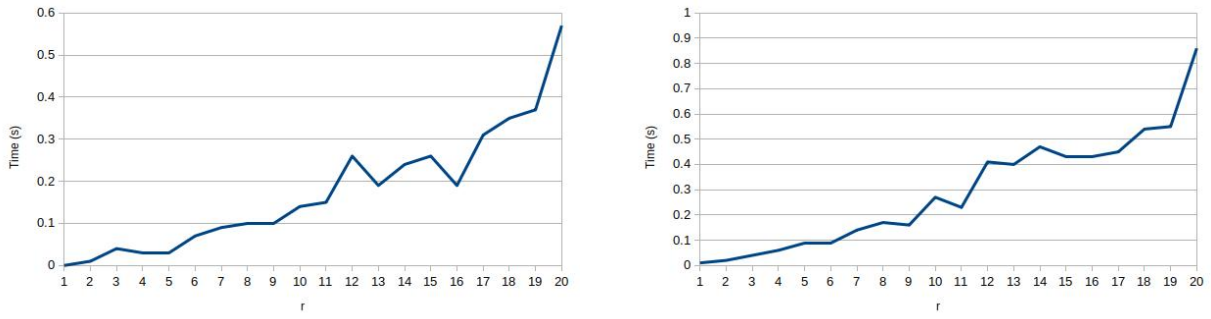
In the experiments, for each dimension $n = 1, \ldots, 50$ and each number of regions $r = 1, \ldots, 20$, one function f was generated with linear piece p_1 defined from a function p for which, for each coefficient $\frac{a_j}{b_j}$, a_j was randomly chosen among integers from -100 to 100 and b_j was 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 3.9, 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



(a) *Simple-region config.* **(b)** *Cubic-region config.*

Figure 3.8: *Simple-region and cubic-region configurations in dimension* $n = 3$



(a) *Dimension* $n = 25$ **(b)** *Dimension* $n = 50$

Figure 3.9: *Representation builder performance running on simple-region piecewise linear functions, randomly gen. instances:* $r = 1$ *to* $r = 20$

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 3.8b 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 (3.6) by the normalization process in (3.16) performed over $D = \Omega_{\langle 1,\ldots,1\rangle}$.

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

$$p_{\langle i_1,\ldots,i_n\rangle}(\mathbf{x}) = p_{\langle i_1,\ldots,i_{j-1},i_j-1,i_{j+1},\ldots,i_n\rangle}(\mathbf{x}) + q_j^{i_j} \cdot \left(x_j - \frac{i_j - 1}{r}\right), \tag{3.17}$$

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

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

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 (3.17) 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 $l \ge k$; indeed, there is a value q such that

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

and, since

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

we are able to write

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

Thus, functions f and f' are continuous over $[0,1]^n$. Theorem 13 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 3.7, we see the results of the representation builder running on cubic-region piecewise linear functions.

Theorem 13 *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 11. Since, by Theorem 12, 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

$$p_{\langle i_1,\ldots,i_n\rangle}(\mathbf{x}) \geq p_{\langle i_1,\ldots,i_{j-1},k_j,i_{j+1},\ldots,i_n\rangle}(\mathbf{x}),$$

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

$$p_{\langle i_1,\ldots,i_n\rangle}(\mathbf{x}) \geq p_{\langle k_1,\ldots,k_n\rangle}(\mathbf{x}), \text{ for } \mathbf{x} \in \Omega_{\langle i_1,\ldots,i_n\rangle}, \text{ and } p_{\langle k_1,\ldots,k_n\rangle}(\mathbf{x}) \geq p_{\langle I_1,\ldots,I_n\rangle}(\mathbf{x}), \text{ for } \mathbf{x} \in \Omega_{\langle I_1,\ldots,I_n\rangle}.$$

Therefore, f has the lattice property. ■

n	r	**Average time (s)**	**Minimum time (s)**	**Maximum time (s)**	n	r	**Average time (s)**	**Minimum time (s)**	**Maximum time (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					

Table 3.7: *Representation builder performance running on cubic-region piecewise linear functions*

3.6 An Application to the Formal Analysis of Neural Networks

Neural networks are computational models that aim to generalize a pattern found in a dataset from which they are modeled; for our purposes, we identify neural networks with the computable functions they determine. For instance, given a dataset with observations on the weather conditions of a day together with the information of whether in the next day it rains or not, we are able to define — or *train*, in the jargon of the field —, based on these data, a neural network that, given today's weather conditions, predicts whether it will rain tomorrow; such defining procedure is done by means of the so-called learning algorithms (Goodfellow *et al.*, 2016). Unfortunately, a drawback in these models — and in automated learning methods in general — is the impossibility to directly inspect the learned information; it is thus desirable to come up with methods to formally analyze some aspects of them, such as reachability of a given state or robustness (Finger, 2020).

Neural networks are notorious examples of functions that may be approximated within the Ł$_\infty$-MODSAT system; in fact, a neural network, depending on its class of activation functions, can be seen either as a piecewise linear function or as a continuous function that can be approximated by one (Leshno *et al.*, 1993). In this way, properties of neural networks might be translated into properties of Ł$_\infty$, which paves the way for their formal verification. We next show some ways to formally verify reachability and robustness via properties of Ł$_\infty$ for the particular cases of neural networks that are exactly rational McNaughton functions.

Let $f : [0,1]^n \to [0,1]$ be a neural network which is a rational McNaughton function and that, according to an input $\mathbf{x} \in [0,1]^n$, induces prediction **Yes**, if $f(\mathbf{x}) \geq 0.5$, and **No**, if $f(\mathbf{x}) < 0.5$ — for instance, for answering the question of whether it will rain tomorrow —; $f(\mathbf{x})$ may be interpreted as the probability of the answer being **Yes**. Unless stated otherwise, whenever we refer to a neural network in this section, we are referring to a neural network of the type we have just defined.

The *reachability* of a given state in our context may be seen as the problem of determining if a neural network $f : [0,1]^n \to [0,1]$ reaches a specific probability $\pi = \frac{a}{b} \in [0,1] \cap \mathbb{Q}$, that is determining whether there is some $\mathbf{x} \in [0,1]^n$ for which $f(\mathbf{x}) = \pi$, or as the problem of determining if f reaches at least probability $\pi = \frac{a}{b} \in [0,1] \cap \mathbb{Q}$, that is determining whether there is some $\mathbf{x} \in [0,1]^n$ for which $f(\mathbf{x}) \geq \pi$. Such properties may be modeled in terms of the satisfiability of a Ł$_\infty$-formula. Let $\langle \varphi, \Phi \rangle$ be a representation of f; we claim that f reaches probability $\pi = \frac{a}{b}$ if, and only if, formula

$$\left(\bigwedge \Phi\right) \wedge \varphi_{\frac{1}{b}} \wedge aZ_{\frac{1}{b}} \leftrightarrow \varphi \tag{3.18}$$

is satisfiable and that f reaches at least probability $\pi = \frac{a}{b}$ if, and only if, formula

$$\left(\bigwedge \Phi\right) \wedge \varphi_{\frac{1}{b}} \wedge aZ_{\frac{1}{b}} \to \varphi \tag{3.19}$$

is satisfiable.

Theorem 14 *Let $f : [0,1]^n \to [0,1]$ be a neural network which is a rational McNaughton function and $\langle \varphi, \Phi \rangle$ be its representation. Then, f reaches (at least) probability $\pi = \frac{a}{b} \in [0,1] \cap \mathbb{Q}$ if, and only if, formula (3.18) (formula (3.19)) is satisfiable.* □

Proof If a valuation $v \in \mathbf{Val}$ satisfies formula (3.18), then $v \in \mathbf{Val}_\Phi$ and $v(Z_{\frac{1}{b}}) = \frac{1}{b}$. Thus, since $v(aZ_{\frac{1}{b}} \leftrightarrow \varphi) = 1$, it follows that $\pi = \frac{a}{b} = f(v(X_1), \ldots, v(X_n))$; i.e. f reaches probability π.

Reciprocally, if f reaches probability π, there is $\mathbf{x} \in [0,1]^n$ for which $f(\mathbf{x}) = \pi$ and, since $\langle \varphi, \Phi \rangle$ represents f, there is a valuation $v \in \mathbf{Val}_\Phi$, such that $\mathbf{x} = \langle v(X_1), \ldots, v(X_n) \rangle$; we may assume without any loss of generality that $v(Z_{\frac{1}{b}}) = \frac{1}{b}$. Therefore, $v(aZ_{\frac{1}{b}}) = \pi = v(\varphi)$ and v satisfies (3.18). The argument is analogous according to formula (3.19) for determining whether f reaches at least probability π. ■

Corollary 1 *The problem of deciding if a neural network given by a rational McNaughton function in regional format reaches (at least) probability $\pi = \frac{a}{b} \in [0,1] \cap \mathbb{Q}$ is in NP.* □

Proof By Theorem 14, these problems may be reduced to the satisfiability of formulas (3.18) and (3.19). By Lemma 6, the formulas

$$\left(\bigwedge \Phi\right) \wedge \left(\bigwedge \Xi_{(b-1)Z_{\frac{1}{b}}}\right) \wedge \left(\bigwedge \Xi_{aZ_{\frac{1}{b}}}\right) \wedge Z_{\frac{1}{b}} \leftrightarrow \neg \xi_{(b-1)Z_{\frac{1}{b}}} \wedge \xi_{aZ_{\frac{1}{b}}} \leftrightarrow \varphi \tag{3.20}$$

and

$$\left(\bigwedge \Phi\right) \wedge \left(\bigwedge \Xi_{(b-1)Z_{\frac{1}{b}}}\right) \wedge \left(\bigwedge \Xi_{aZ_{\frac{1}{b}}}\right) \wedge Z_{\frac{1}{b}} \leftrightarrow \neg \xi_{(b-1)Z_{\frac{1}{b}}} \wedge \xi_{aZ_{\frac{1}{b}}} \rightarrow \varphi \tag{3.21}$$

are satisfiable if, and only if, formulas (3.18) and (3.19) are respectively satisfiable. By Theorem 10 and by the observations about Algorithms 2 and 3, the above formulas may be computed in polynomial time from a neural network given by a rational McNaughton function in regional format and a given probability π. The result follows as Ł$_\infty$-SAT is a problem in NP (Mundici, 1987). ■

Algorithm 6 MaxProbability-BS: Computes value Π via Binary Search

Input: A neural network which is a rational McNaughton function and a precision $\delta > 0$.
Output: Value Π with precision δ.

```
Build expression Λ(1/1);
if Λ(1/1) is satisfiable then
        v_min := 1;
else
        k := ⌈|log δ|⌉;
        j := 1, v_min := 0;
        while j ≤ k do
                v_max := v_min + 1/2^j;
                Build expression Λ(v_max);
                if Λ(v_max) is satisfiable then
                        v_min := v_max;
                end if
                j++;
        end while
end if
return v_min;
```

It is possible to determine, according to a precision $\delta = 2^{-k}$, the maximum probability Π which a neural network reaches by means of a binary search through the possible values in the binary representation of Π. Algorithm 6 presents such procedure; $\Lambda(\pi)$ stands for (3.19) or (3.21) with highlighted value π. First iteration consists of verifying if $\Lambda(\frac{1}{1})$ is satisfiable; if it is, $\Pi = 1$, if not,

$\Pi = 0$ with precision $2^0 = 1$. In case the former iteration was unsatisfiable, the second iteration consists of verifying if $\Lambda(\frac{1}{2})$ is satisfiable; if it is the case, $\Pi = \frac{1}{2}$, if it is not, $\Pi = 0$, both cases with precision $2^{-1} = \frac{1}{2}$. One more iteration will give precision $2^{-2} = \frac{1}{4}$ and it consists of verifying the satisfiability of $\Lambda(\frac{3}{4})$ in case the former iteration had positive answer, or of $\Lambda(\frac{1}{4})$ in case it was negative. The process continues until the precision desired and it takes $|\log \delta| + 1 = k + 1$ iterations to be completed.

Theorem 15 *Given a precision $\delta > 0$, maximum probability Π that a neural network which is a rational McNaughton function reaches may be computed with $O(|\log \delta|)$ checks of $\Lambda(\pi)$ in (3.19) or (3.21).* □

Neural networks in general are said to be *robust* if they maintain their predictions even when inputs are stricken by small perturbations. The literature provides examples of deep neural networks (that are not necessarily rational McNaughton functions) which perform well at the task of image recognition and, nonetheless, are susceptible to *adversarial examples*, that is, images with small perturbations which were originally classified properly and, still, had the prediction changed after the perturbation (Szegedy *et al.*, 2014).

We want to verify whether a neural network which is a rational McNaughton function is robust when predicting **Yes** with respect to a fixed "small" perturbation limit $\varepsilon = \frac{\alpha}{\beta} \in \mathbb{Q}$ and to a "big" probability $\pi = \frac{a}{b} \in [0,1] \cap \mathbb{Q}$ such that $\pi \geq \frac{1}{2}$, that is whether $f(\mathbf{x} + \mathbf{p}) \geq 0.5$, for all $\mathbf{x} \in [0,1]^n$ and $\mathbf{p} = \langle p_1, \ldots, p_n \rangle \in \mathbb{R}^n$ such that $f(\mathbf{x}) \geq \pi$, $|p_i| \leq \varepsilon$, for $i = 1, \ldots, n$, and $\mathbf{x} + \mathbf{p} \in [0,1]^n$. We model such concept of robustness by an instance of logical consequence in Ł_∞. Let $\langle \varphi, \Phi \rangle$ be a representation of f for which $\mathbf{X}_n$ determines φ modulo Φ-satisfiable; for each propositional variable $X \in \text{Var}(\varphi) \cup \text{Var}(\Phi)$, we introduce a new variable X' and for each propositional variable $X_i \in \mathbf{X}_n$, we introduce a new variable P_i. Define the pair $\langle \varphi', \Phi' \rangle$, where all occurrences of variables X in φ or Φ are replaced by X' in order to obtain φ' and Φ'. We claim that robustness of f with respect to perturbation limit $\varepsilon = \frac{\alpha}{\beta}$ and to probability $\pi = \frac{a}{b}$ is equivalent to the validity of the logical consequence

$$\begin{aligned} \Phi,\ \Phi',\ \varphi_{\frac{1}{\beta}},\ \varphi_{\frac{1}{b}},\ \varphi_{\frac{1}{2}}, \\ P_1 \to \alpha Z_{\frac{1}{\beta}},\ \ \ldots,\ P_n \to \alpha Z_{\frac{1}{\beta}}, \\ (X_1' \leftrightarrow X_1 \oplus P_1) \vee (X_1' \leftrightarrow \neg(X_1 \to P_1)), \\ \ldots, \\ (X_n' \leftrightarrow X_n \oplus P_n) \vee (X_n' \leftrightarrow \neg(X_n \to P_n)), \\ aZ_{\frac{1}{b}} \to \varphi \quad \models_{\text{Ł}_\infty} Z_{\frac{1}{2}} \to \varphi'. \end{aligned} \tag{3.22}$$

Theorem 16 *Let $f : [0,1]^n \to [0,1]$ be a neural network which is a rational McNaughton function and $\langle \varphi, \Phi \rangle$ be its representation from which $\langle \varphi', \Phi' \rangle$ is defined as in previous discussion. Then, f is robust with respect to $\varepsilon = \frac{\alpha}{\beta} \in \mathbb{Q}$ and $\pi = \frac{a}{b} \in [0,1] \cap \mathbb{Q}$ if, and only if, (3.22) holds.* □

Proof First note that letting X_i' play the role of X_i, for $i = 1, \ldots, n$, $\langle \varphi', \Phi' \rangle$ represents f by construction. Assume f is robust and let $v \in \mathbf{Val}$ satisfy all premises in (3.22); in particular, $v \in \mathbf{Val}_\Phi \cap \mathbf{Val}_{\Phi'}$. Let $x_i = v(X_i)$ and

$$p_i = \begin{cases} \min\{v(P_i), 1 - v(X_i)\}, & \text{if } v(X_i' \leftrightarrow X_i \oplus P_i) = 1 \\ -\min\{v(P_i), v(X_i)\}, & \text{if } v(X_i' \leftrightarrow \neg(X_i \to P_i)) = 1 \end{cases}$$

for $i = 1, \ldots, n$; then, $\mathbf{x}+\mathbf{p} \in [0,1]^n$, $|p_i| \leq v(P_i) \leq v(\alpha Z_{\frac{1}{\beta}}) \leq \varepsilon$, for $i = 1, \ldots, n$, and $\pi = v(aZ_{\frac{1}{b}}) \leq v(\varphi) = f(\mathbf{x})$. Since $v(X'_i) = x_i + p_i$, for $i = 1, \ldots, n$, by the robustness of f, we have that

$$v(Z_{\frac{1}{2}}) = 0.5 \leq f(\mathbf{x}+\mathbf{p}) = f(v(X'_1), \ldots, v(X'_n)) = v(\varphi').$$

It follows that (3.22) holds. On the other hand, assume that (3.22) holds and let $\mathbf{x} \in [0,1]^n$ and $\mathbf{p} \in \mathbb{R}^n$ be such that $\mathbf{x}+\mathbf{p} \in [0,1]^n$, $|p_i| \leq \varepsilon$, for $i = 1, \ldots, n$, and $f(\mathbf{x}) \geq \pi$. It is easy to see that there is a valuation $v \in \mathbf{Val}_\Phi \cap \mathbf{Val}_{\Phi'}$ such that $v(X_i) = x_i$ and $v(X'_i) = x_i + p_i$, for $i = 1, \ldots, n$, and, also, such that $v(\varphi_{\frac{1}{\beta}}) = v(\varphi_{\frac{1}{b}}) = v(\varphi_{\frac{1}{2}}) = 1$ and $v(P_i) = |p_i|$, for $i = 1, \ldots, n$. By the assumptions on $\mathbf{x}$ and $\mathbf{p}$, we have that $v(P_i \to \alpha Z_{\frac{1}{\beta}}) = v((X'_i \leftrightarrow X_i \oplus P_i) \vee (X'_i \leftrightarrow \neg(X_i \to P_i))) = 1$, for $i = 1, \ldots, n$, and $v(aZ_{\frac{1}{b}} \to \varphi) = 1$. Therefore, $0.5 = v(Z_{\frac{1}{2}}) \leq v(\varphi') = f(\mathbf{x}+\mathbf{p})$ and f is robust. ■

Corollary 2 *The problem of deciding if a neural network given by a rational McNaughton function in regional format is robust with respect to $\varepsilon = \frac{\alpha}{\beta} \in \mathbb{Q}$ and $\pi = \frac{a}{b} \in [0,1] \cap \mathbb{Q}$ is in coNP.* □

Proof First dealing with the possibly exponential formulas $\varphi_{\frac{1}{\beta}}$, $\varphi_{\frac{1}{b}}$, $\alpha Z_{\frac{1}{\beta}}$ and $aZ_{\frac{1}{b}}$ through Lemma 6, similarly as done in the proof of Corollary 1, the result follows as the decision of validity of logical consequences in $\mathrm{Ł}_\infty$ is a problem in coNP (Aguzzoli and Ciabattoni, 2000, Theorem 17). ■

Algorithm 7 MaxPerturb-BS: Computes value E via Binary Search

Input: A neural network which is a rational McNaughton function, a probability $\pi \geq \frac{1}{2}$ and a precision $\delta > 0$.
Output: Value E with precision δ.

```
Build expression Γ(1/1);
if Γ(1/1) holds then
        v_min := 1;
else
        k := ⌈|log δ|⌉;
        j := 1, v_min := 0;
        while j ≤ k do
                v_max := v_min + 1/2^j;
                Build expression Γ(v_max);
                if Γ(v_max) holds then
                        v_min := v_max;
                end if
                j++;
        end while
end if
return v_min;
```

Again, we can determine, according to a precision $\delta = 2^{-k}$, the maximum perturbation limit E for which a neural network remains robust with respect to a fixed probability π by means of a binary search through the possible values in the binary representation of E. Algorithm 7 presents such procedure; $\Gamma(\varepsilon)$ stands for (3.22) with highlighted value ε. Analogous to Algorithm 6, the iterations in Algorithm 7 verify if an expression $\Gamma(\varepsilon)$ holds and each new iteration refines the precision of value E.

Theorem 17 *Given a precision $\delta > 0$, the maximum perturbation limit E of a neural network which is a rational McNaughton function with respect to a fixed probability π may be computed with $O(|\log \delta|)$ checks of $\Gamma(\varepsilon)$ in (3.22).* ∎

The burden in Algorithms 6 and 7 falls on verifying properties of Ł_∞ — satisfiability and logical consequence — and on building logical expressions $\Lambda(\pi)$ and $\Gamma(\varepsilon)$. Nevertheless, the biggest role in these expressions is played by representation $\langle \varphi, \Phi \rangle$ of a rational McNaughton function f, whose building procedure is studied in Sections 3.4.2 and 3.4.3, that are devoted to efficiently performing such task.

3.6.1 Verifying a Rain Forecast Neural Network

As an experiment, we actually perform formal verification of a neural network that predicts *whether it will or not rain tomorrow in Australia*, given today's *rainfall*, *humidity at 3:00 PM*, *pressure at 9:00 AM* and *whether it rained today*. These four latter separate data — usually called features — are codified as entries in a tuple $\langle x_1, x_2, x_3, x_4 \rangle \in [0,1]^4$ in order to constitute the input for our neural network.

Given measures $\bar{x}_1$, $\bar{x}_2$ and $\bar{x}_3$ of rainfall, humidity and pressure, respectively — which we assume to be between $\min(\bar{x}_j)$ and $\max(\bar{x}_j)$, the minimum and maximum values in the training data —, we calculate

$$x_j = \frac{\bar{x}_j - \min(\bar{x}_j)}{\max(\bar{x}_j) - \min(\bar{x}_j)} \in [0,1],$$

for $j = 1, 2, 3$; whether it rained today is already a binary value $x_4 \in \{0, 1\}$. The output of our neural network $f_R(x_1, x_2, x_3, x_4) \in [0,1]$ is interpreted as the probability of raining tomorrow and induces answer **Yes**, for values at least 0.5, and answer **No**, otherwise.

We trained a feedforward neural network $f_R : [0,1]^4 \to [0,1]$ with one hidden layer with four nodes, that means function f_R is given by

$$f_R(\mathbf{x}) = \tau\Big(o\big(\ \rho\big(h_1(\mathbf{x})\big),\ \ \rho\big(h_2(\mathbf{x})\big),\ \ \rho\big(h_3(\mathbf{x})\big),\ \ \rho\big(h_4(\mathbf{x})\big)\ \big)\Big),$$

where the hidden layer is composed of linear functions $h_j : [0,1]^4 \to \mathbb{R}$, for $j = 1, \ldots, 4$, which are transformed by the rectified linear unit function $\rho : \mathbb{R} \to \mathbb{R}$, given by $\rho(x) = \max(0, x)$; the output layer is composed of a linear function $o : \mathbb{R}^4 \to \mathbb{R}$, which is transformed by the truncation function $\tau : \mathbb{R} \to [0,1]$, given by $\tau(x) = \min(1, \max(0, x))$. Figure 3.10 graphically depicts the neural network. The training of the neural network[2] consists exactly of the definition of the parameters in functions h_j, for $j = 1, \ldots, 4$.

The training data we used comprehends daily observations of about ten years in several cities in Australia[3] on rainfall, humidity at 3:00 PM, pressure at 9:00 AM, whether it rained that day and whether it rained in the next day, which is also given by a binary variable and is the information our neural network is intended to predict, called the target variable.

[2]The neural network we discuss was trained using the PyTorch 1.5.0 library in Python 3.7.3 with the optimizer Adam, optimization criterion BCELoss and learning rate 0.0001.

[3]These data were curated by Joe Young and are publicly available at www.kaggle.com/jsphyg/weather-dataset-rattle-package.

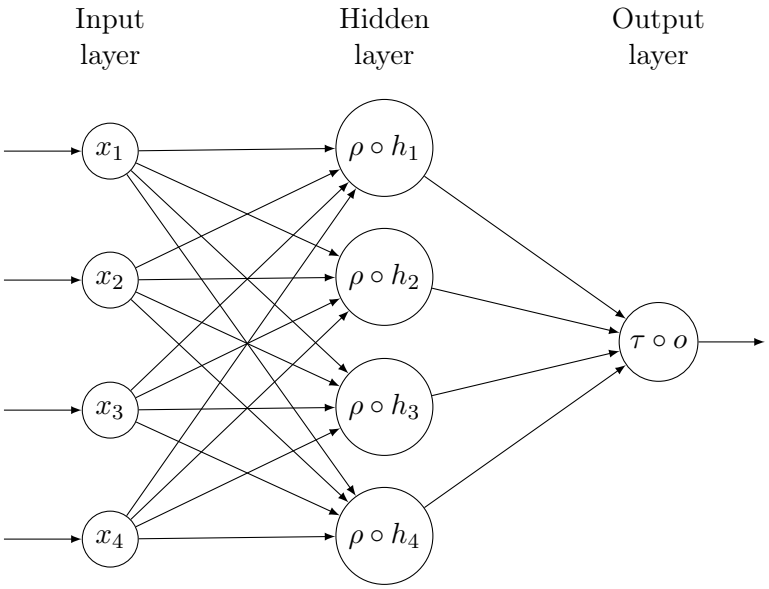


Figure 3.10: *Graphical representation of neural network f_R*

Neural network f_R is exactly a rational McNaughton function and may be encoded in pre-regional format with 48 regions given by

$$\Omega_{\langle i_1,i_2,i_3,i_4,\omega\rangle} = \Big\{\mathbf{x} \in [0,1]^4 \;\Big|\; h_j(\mathbf{x}) \bowtie_j 0, \text{ for } j = 1,\ldots,4\Big\} \cap \Omega'_{\langle i_1,i_2,i_3,i_4,\omega\rangle},$$

for $i_1, i_2, i_3, i_4 \in \{0,1\}$ and $\omega \in \{0,1,2\}$, where $\bowtie_j$ is the symbol $\leq$, if $i_j = 0$, and the symbol $\geq$, if $i_j = 1$. Let

$$\chi_j(\mathbf{x}) = \begin{cases} 0, & \text{if } i_j = 0 \\ h_j(\mathbf{x}), & \text{if } i_j = 1 \end{cases};$$

for $\omega \in \{0,1\}$, define

$$\Omega'_{\langle i_1,i_2,i_3,i_4,\omega\rangle} = \Big\{\mathbf{x} \in [0,1]^4 \;\Big|\; o\big(\, \chi_1(\mathbf{x}),\ \chi_2(\mathbf{x}),\ \chi_3(\mathbf{x}),\ \chi_4(\mathbf{x}) \,\big) \bowtie \omega\Big\},$$

where $\bowtie$ is the symbol $\leq$, if $\omega = 0$, and the symbol $\geq$, if $\omega = 1$; for $\omega = 2$, define

$$\Omega'_{\langle i_1,i_2,i_3,i_4,\omega\rangle} = \Big\{\mathbf{x} \in [0,1]^4 \;\Big|\; 0 \leq o\big(\, \chi_1(\mathbf{x}),\ \chi_2(\mathbf{x}),\ \chi_3(\mathbf{x}),\ \chi_4(\mathbf{x}) \,\big) \leq 1\Big\}.$$

For $\mathbf{x} \in \Omega_{\langle i_1,i_2,i_3,i_4,\omega\rangle}$,

$$f_R(\mathbf{x}) = p_{\langle i_1,i_2,i_3,i_4,\omega\rangle}(\mathbf{x}) = \begin{cases} 0, & \text{if } \omega = 0 \\ 1, & \text{if } \omega = 1 \\ o\big(\, \chi_1(\mathbf{x}),\ \chi_2(\mathbf{x}),\ \chi_3(\mathbf{x}),\ \chi_4(\mathbf{x}) \,\big), & \text{if } \omega = 2 \end{cases}.$$

All regions in this encoding are closed, so we were able to verify that it has the lattice property by means of exhaustive searches where decisions about whether a linear piece is above another one over some polyhedron were made by a routine based on Algorithm 1. Therefore, the encoding is actually in closed regional format.

Property	Precision	Maximum value
Reachability	$\delta = 2^{-5}$	$\Pi \approx \frac{17}{32} = 0.53125$
Reachability	$\delta = 2^{-10}$	$\Pi \approx \frac{275}{512} = 0.537109375$
Robustness for $\pi = \frac{1}{2}$	$\delta = 2^{-5}$	$E \approx 0$
Robustness for $\pi = \frac{51}{100}$	$\delta = 2^{-5}$	$E \approx 0$
Robustness for $\pi = \frac{52}{100}$	$\delta = 2^{-5}$	$E \approx \frac{1}{32}$
Robustness for $\pi = \frac{53}{100}$	$\delta = 2^{-5}$	$E \approx \frac{1}{32}$
Robustness for $\pi = \frac{17}{32}$	$\delta = 2^{-5}$	$E \approx \frac{1}{32}$
Robustness for $\pi = \frac{275}{512}$	$\delta = 2^{-5}$	$E \approx \frac{1}{16}$

Table 3.8: *Properties of neural network f_R inferred through formal verification methods*

Using Algorithms 6 and 7, we were able to determine properties of the neural network f_R; its representation in Ł$_\infty$-MODSAT was built by means of our implementation of Algorithm 5 (Section 3.5). Verification of satisfiability and logical consequence in Ł$_\infty$ were made by a Ł$_\infty$-solver and a Ł$_\infty$-theorem prover 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). This experiment was run on a shared UNIX machine with two E5645 CPUs @ 2.40GHz with 12 processors. Although the real elapsed time for building the representation of f_R in Ł$_\infty$-MODSAT was less than one second, for robustness verifications it was up to two months; such performance may be credited to the lack of known suitable techniques for deciding on the validity of logical consequence in Ł$_\infty$. The results of our experiments are summarized in Table 3.8 and the source code for them is publicly available.[4]

3.7 Modulo Satisfiability versus Traditional Representation

The representation modulo satisfiability has as important motivation the advantageous time complexity involved in building representations, which is polynomial time for rational McNaughton functions in regional format as showed in this chapter, and in the tasks related to Ł$_\infty$, for which, for instance, Ł$_\infty$-SAT is NP-complete as showed by Mundici (1987). More than that, Ł$_\infty$-solvers have already been studied to the point of having the phenomenon of phase transition established (Bofill *et al.*, 2015).

On the other hand, there are propositional logics whose formulas represent rational McNaughton functions in the traditional way as in Section 3.1. In the following, we present and discuss some of the most relevant traditional approaches.

- Logic Ł$\Pi\frac{1}{2}$ extends Ł$_\infty$ with a product operator, its residuum and a constant expressing the truth value $\frac{1}{2}$, not directly expressible in Ł$_\infty$ (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 Ł$\Pi\frac{1}{2}$ requires finding roots of polynomials of n-degree making its complexity extremely high.

[4] http://github.com/spreto/NNverificationViaLukasiewiczLogic

- Logic ∃Ł also expresses rational McNaughton functions (Aguzzoli and Mundici, 2001, 2003); it extends $Ł_\infty$ and introduces rational numbers by providing a restricted form of propositional quantification whose semantic counterpart is the maximization of a set of $Ł_\infty$-valuations of a formula. The satisfiability problem in that logic is in the complexity class Σ_2^p, which is also a high complexity.

- Rational Łukasiewicz Logic extends $Ł_\infty$ 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, however there is no known algorithm to build the representative formulas and an attempt to derive one from the results of Gerla (2001) would lead to the problem of representing McNaughton functions in $Ł_\infty$; it is known that this task may be done in polynomial time on the coefficients of some specific functions (Aguzzoli, 1998), however these methods lead to exponential time complexity if binary representation of the coefficients is used.

- Logic RŁ extends $Ł_\infty$ with constant multiplication operators ∇_r that induce 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 $q \in [0,1] \cap \mathbb{Q}$.

Let us explore the connections between representation modulo satisfiability and Rational Łukasiewicz Logic. The McNaughton-like theorem in the work of Gerla (2001) establishes a one-to-one correspondence between equivalence classes modulo equi-provability of formulas of Rational Łukasiewicz Logic and rational McNaughton functions. According to this result, a rational McNaughton function $f : [0,1]^n \to [0,1]$ is represented by a class of (equi-provable) formulas of Rational Łukasiewicz Logic which has among them the formula in special format

$$\varphi = \bigoplus_{i=0}^{s-1} \delta_s \varphi_i, \tag{3.23}$$

where s is some integer for which the linear pieces of $s \cdot f$ have integer coefficients and φ_i are representations in $Ł_\infty$ for the McNaughton functions $f_i : [0,1]^n \to [0,1]$ given, for $\mathbf{x} \in [0,1]^n$, by

$$f_i(\mathbf{x}) = \max\Big(\min\big(s \cdot f(\mathbf{x}) - i, 1\big), 0\Big).$$

In a sense, the following result says that operators δ_n may be represented in $Ł_\infty$-MODSAT.

Theorem 18 *Let φ be a formula with $Var(\varphi) \subseteq \mathbf{X}_n$ and $s \in \mathbb{N}^*$. Then, function*

$$\frac{1}{s} \cdot f_\varphi : \mathbf{x} \in [0,1]^n \mapsto \frac{f_\varphi(\mathbf{x})}{s}$$

is representable in $Ł_\infty$-MODSAT. □

PROOF With new variables W and $Z_{\frac{1}{s}}$, we define

$$\Phi = \left\{ \varphi_{\frac{1}{s}}, \quad sW \leftrightarrow \varphi, \quad W \to Z_{\frac{1}{s}} \right\}$$

and, analogous to the proofs of Lemmas 2 and 5, we have that $\langle W, \Phi \rangle$ represents $\frac{1}{s} \cdot f_\varphi$. ∎

By Theorem 18 and special format (3.23), we may say that any class of equi-provable formulas of Rational Łukasiewicz Logic is representable in Ł$_\infty$-MODSAT: let φ be the formula in such class as in (3.23), then the representation is given by the pair $\langle \varphi, \Phi \rangle$, where

$$\varphi = \bigoplus_{i=0}^{s-1} W_i$$

and

$$\Phi = \left\{ \varphi_{\frac{1}{s}} \right\} \quad \cup \quad \bigcup_{i=0}^{s-1} \left\{ sW_i \leftrightarrow \varphi_i, \quad W_i \to Z_{\frac{1}{s}} \right\}.$$

Of course, since such classes are identified with rational McNaughton functions, that was already a consequence of Theorem 3.

Part of this chapter's results has appeared in the following publications.

- Preto and Finger (2020) Sandro Preto and Marcelo Finger. An efficient algorithm for representing piecewise linear functions into logic. *Electronic Notes in Theoretical Computer Science*, 351:167-186. ISSN 1571-0661. doi: 10.1016/j.entcs.2020.08.009. URL http://doi.org/10.1016/j.entcs.2020.08.009. Proceedings of LSFA 2020, the 15th International Workshop on Logical and Semantic Frameworks, with Applications (LSFA 2020).

- Finger and Preto (2020) Marcelo Finger and Sandro Preto. Probably partially true: Satisfiability for Łukasiewicz infinitely-valued probabilistic logic and related topics. *Journal of Automated Reasoning*, 64(7):1269-1286. ISSN 1573-0670. doi: 10.1007/s10817-020-09558-9. URL http://doi.org/10.1007/s10817-020-09558-9.

- Preto and Finger (2019) Sandro Preto and Marcelo Finger. Representing rational McNaughton functions via MODSAT relativisation. In Cezar Augusto Mortari, Ricardo Silvestre, Ítala Maria Loffredo D'Ottaviano, Leandro Suguitani and Petrucio Viana, editors, *19th Brazilian Logic Conference EBL 2019: Book of Abstracts*, page 183. Mídia Gráfica e Editora Ltda, UFCG-EDUFCG.

Furthermore, the following papers, which are also related to this chapter, have been submitted to journals and are currently under review.

- Sandro Preto and Marcelo Finger. Efficient representation of piecewise linear functions into Łukasiewicz logic modulo satisfiability.

- Sandro Preto and Marcelo Finger. Proving properties of binary classification neural networks via Łukasiewicz logic.

Chapter 4

Probabilities over Łukasiewicz Infinitely-Valued Logic

There are situations where, instead of classical truth values, a gradation of truth may be closer to the perceptions of the agents involved. More than that, by departing from the classical probabilistic setting and instead employing Łukasiewicz Infinitely-valued Logic as underlying logic, we enlarge our probabilistic reasoning capacity in order to comprehend such situations. A sound probability theory over such a many-valued context that includes a notion of coherent probabilities in line with de Finetti's was developed by Mundici (2006). The following example illustrates such setting.

Example 6 Three friends have the habit of going to a bar to watch their soccer team's matches. Staff at the bar claims that at every such match at least two of the friends come to the premises, but if you ask them, they will say that each of them comes to watch at most 60% of the games.

In classical terms, the claims of the staff and of the three friends are in contradiction. In fact, if there are always two of the three friends present at matches, someone must attend to least two-thirds of the team's matches.

However, one may allow someone to arrive for the second half of the match and consider his attendance only "partially true",[1] say, a truth value of 0.5 in that case. Then it may well be the case that staff and customers are both telling the truth, that is, their claims are jointly satisfiable. ■

Despite the above example being unsatisfiable according to classical probability theory, it is satisfiable in a probability theory grounded on Łukasiewicz Infinitely-valued Logic. In this chapter, we deal with the problem of deciding whether a set of probabilistic assignments over $Ł_\infty$ is coherent, the ŁIPSAT problem. Our goal is to explore equivalent formulations and algorithmic ways to solve it and, as it is an NP-complete problem (Bova and Flaminio, 2010), study the existence of a phase transition in the empirical behavior of such solutions.

We propose a ŁIPSAT-solving algorithm where semantics modulo satisfiability is combined with techniques from linear programming in order to solve instances given in a normal form which, analogous to the classical case in Section 2.3, represents probabilistic assignments in agreement with a $Ł_\infty$-propositional theory. This algorithm needs to solve several instances of $Ł_\infty$-SAT, for which there are some implementations discussed in the literature (Bofill *et al.*, 2015), but there are many implementation options with considerable efficiency differences which we analyze.

[1] See footnote no. 1, p. 10.

In Section 4.1 we establish a non-classical probability theory grounded on Ł_∞ and its corresponding notion of coherent probability assignment. In Section 4.2 we study the theoretical relationship between linear algebraic methods, semantics modulo satisfiability and the solution of the ŁIPSAT problem. In Section 4.3 we develop a column generation algorithm for ŁIPSAT-solving and show its correctness. Finally, we discuss implementation issues and the phase transition behavior of the solvers in Section 4.4.

Throughout this chapter, we refer to Ł_∞-formulas, Ł_∞-valuations, Ł_∞-partial valuations and the semantics Ł_∞-**Val** simply as formulas, valuations, partial valuations and **Val**.

4.1 Łukasiewicz Probabilities and Coherence

The same way Classical Propositional Logic serves as basis for classical probability theory (see Section 2.3), Łukasiewicz Infinitely-valued Logic serves as basis for a non-classical probability theory (Mundici, 2011). Fix a set of propositional variables $\mathcal{P} \subseteq \mathbb{P}$ and define a *convex combination* over a finite set of partial valuations $v_1, \ldots, v_m \in \mathbf{Val}^{\mathcal{P}}$ as a function on formulas φ, with $\text{Var}(\varphi) \subseteq \mathcal{P}$, into $[0, 1]$ such that

$$C(\varphi) = \lambda_1 v_1(\varphi) + \cdots + \lambda_m v_m(\varphi), \tag{4.1}$$

where $\lambda_i \geq 0$ and $\sum_{i=1}^{m} \lambda_i = 1$. So a Ł_∞-probability distribution $\lambda = [\lambda_1, \cdots, \lambda_m]'$ is a vector of coefficients that form the convex combination of partial valuations. To distinguish Ł_∞-probabilities from classical ones, we use the notation $C(\cdot)$, following Mundici (2011); note that classical discrete probabilities are also convex combinations of CPL-partial valuations.

This notion of probability is intrinsically discrete and associates nonzero values only to a finite number of partial valuations, the remaining ones being assumed to have value zero.[2] Since we are interested in deciding the existence of convex combinations of the form (4.1), given a set of constraints, and there are infinitely many possible partial valuations, the search space is *a priori* infinite.

It follows immediately from the definition that $C(\varphi) = 1$ if the underlying convex combination over $v_1, \ldots, v_m$ is such that $v_i(\varphi) = 1$, $1 \leq i \leq m$.

Lemma 8 *$C(\alpha \to \beta) = 1$ iff $C(\alpha) \leq C(\beta)$.* □

Proof From the fact that $v(\varphi \to \psi) = 1$ iff $v(\varphi) \leq v(\psi)$. ■

Now, define a Łukasiewicz Infinitely-valued Probabilistic (ŁIP) assignment as an expression of the form

$$\Sigma = \Big\{ C(\alpha_i) = q_i \;\Big|\; q_i \in [0, 1] \cap \mathbb{Q},\ 1 \leq i \leq k \Big\}. \tag{4.2}$$

Since we are concerned with computational problems, we consider only probabilities in $[0, 1] \cap \mathbb{Q}$.

As a foundational view of probabilities, it is possible to define a coherence criterion over ŁIP-assignments, in analogy to the classical notion of coherent assignment of probabilities due to

[2] Such notion is thus more restrictive than the full class of states of an MV-algebra, in the sense of Mundici (2011), which we do not discuss in this work.

de Finetti (1931, 1937, 2017). Thus, we define the Ł_∞-coherence of a ŁIP-assignment $\{C(\alpha_i) = q_i \mid 1 \leq i \leq k\}$ in terms of a bet between two players, Alice the bookmaker and Bob the bettor; the outcome on which the players bet is a partial valuation $v \in \mathbf{Val}^{\mathcal{P}}$, with $\text{Var}(\alpha_i) \subseteq \mathcal{P}$, $i \leq 1 \leq k$, describing an actual "possible world". For each formula α_i, Alice states her betting odd $C(\alpha_i) = q_i \in [0,1]$ and Bob chooses a "stake" $\sigma_i \in \mathbb{Q}$; Bob pays Alice $\sum_{i=1}^{k} \sigma_i \cdot C(\alpha_i)$ with the promise that Alice will pay back $\sum_{i=1}^{k} \sigma_i \cdot v(\alpha_i)$ if the outcome is the partial valuation (or "possible world") v. As in the classical case, the chosen stake σ_i is allowed to be negative, in which case Alice pays Bob $|\sigma_i| \cdot C(\alpha_i)$ and gets back $|\sigma_i| \cdot v(\alpha_i)$ if the world turns out to be v. Alice's total balance in the bet is

$$\sum_{i=1}^{k} \sigma_i (C(\alpha_i) - v(\alpha_i)).$$

We say that there is a *ŁIP-Dutch Book* against Alice's ŁIP-assignment if there is a choice of stakes σ_i such that, for every possible outcome v, Alice's total balance is always negative, indicating a bad choice of betting odds made by Alice.

Definition 5 Given a probability assignment to propositional formulas $\{C(\alpha_i) = q_i \mid 1 \leq i \leq k\}$, the ŁIP-assignment is *coherent* if there are no Dutch Books against it. □

While the coherence of an assignment provides a foundational view to deal with Ł_∞-probabilities, a more computational view is possible, based on the satisfiability of assignments. Such a view will allow a more operational way of dealing with Ł_∞-probabilistic assignments.

Definition 6 A ŁIP-assignment is *satisfiable* if there exists a convex combination C over a finite set of partial valuations that jointly verifies all restrictions in it. □

Example 7 Consider again Example 6, let X_1, X_2, X_3 be variables representing the presence at the bar of each of the three friends. The probabilistic constraint expressing that each friend comes at most 60% of the games can be expressed as

$$C(X_1) = C(X_2) = C(X_3) \leq 0.6, \tag{*}$$

and the fact that at least two of them are present is expressed by the constraints

$$C(X_1 \oplus X_2) = C(X_1 \oplus X_3) = C(X_2 \oplus X_3) = 1 \tag{**}$$

which means that no two of them are simultaneously absent. There are infinitely many ways of obtaining a convex combination of partial valuations that satisfy all six conditions, the simplest of which is achieved with a single partial valuation v, $v(X_1) = v(X_2) = v(X_3) = 0.6$; in fact, $v(X_1 \oplus X_2) = v(X_1 \oplus X_3) = v(X_2 \oplus X_3) = 1$, so we can attribute 100% of probability mass to v.

A similar result can be obtained with three "classical" partial valuations $v_i(X_i) = 0, v_i(X_j) = v_i(X_k) = 1$, for pairwise distinct $i, j, k \in \{1, 2, 3\}$ and a fourth partial valuation $v_4(X_1) = v_4(X_2) = v_4(X_3) = 0.5$. Note all four partial valuations satisfy the formulas in (**). The Ł_∞-probability distribution that assigns 0.2 to v_1, v_2, v_3 and 0.4 to v_4 satisfies all constraints (*) and (**). □

The following result is the characterization of coherence for ŁIP-assignments.

Proposition 3 (Mundici (2006)) *Given a ŁIP-assignment* $\Sigma = \{C(\alpha_i) = q_i \mid 1 \leq i \leq k\}$*, the following are equivalent:*

(a) Σ *is a coherent ŁIP-assignment.*

(b) Σ *is a satisfiable ŁIP-assignment.* □

Proposition 3 asserts that deciding ŁIP-coherence is the same as determining ŁIP-assignment satisfiability — we denote such problem by ŁIPSAT. This result is the Ł_∞ analogous to de Finetti's characterization of coherence of classical probabilistic assignment as equivalent to the probabilistic satisfiability of the assignment (PSAT), which was shown to be an NP-complete problem that can be solved using linear algebraic methods (Georgakopoulos *et al.*, 1988; Nilsson, 1986). It has also been shown by Bova and Flaminio (2010) that deciding the coherence of ŁIP-assignments is an NP-complete problem.

Our goal is to explore efficient ways to decide the coherence of ŁIP-assignments. In analogy to the algorithms used for deciding PSAT (Finger and Bona, 2011; Finger and De Bona, 2015), we explore a linear algebraic formulation of the problem based on semantics modulo satisfiability.

4.2 Algebraic Formulation of ŁIPSAT

We consider an *extended* version of ŁIP-assignments of the form

$$\Sigma = \Big\{ C(\alpha_i) \bowtie_i q_i \;\Big|\; q_i \in [0,1] \cap \mathbb{Q},\;\; \bowtie_i \in \{=, \leq, \geq\},\;\; 1 \leq i \leq k \Big\}. \tag{4.3}$$

Extended ŁIP-assignments may have both inequalities and equalities. Such an assignment is satisfiable if there is a Ł_∞-probability distribution λ that verifies all inequalities and equalities in it. Given an extended ŁIP-assignment $\Sigma = \{C(\alpha_i) \bowtie_i q_i\}$, let $q = [q_1, \ldots, q_k]'$ be the vector of probabilities in Σ and $\bowtie$ the "vector" of (in)equality symbols. Suppose we are given partial valuations $v_1, \ldots, v_m \in \mathbf{Val}^{\mathcal{P}}$, with $\text{Var}(\alpha_i) \subseteq \mathcal{P}$, $1 \leq i \leq k$, and let $\lambda = [\lambda_1, \ldots, \lambda_m]'$ be a vector of convex weights; consider the $k \times m$ matrix $A = [a_{ij}]$ where $a_{ij} = v_j(\alpha_i)$. Then, an extended ŁIP-assignment of the form (4.3) is satisfiable if there are $v_1, \ldots, v_m$ and λ such that the set of algebraic constrains (4.4) has a solution.

$$\begin{aligned} A \cdot \lambda &\bowtie q \\ \textstyle\sum \lambda_j &= 1 \\ \lambda &\geq 0 \end{aligned} \tag{4.4}$$

The condition $\sum \lambda_j = 1$ can be incorporated as an all-1 row $k+1$ in matrix A, $q = [q_1, \ldots, q_k, 1]'$ and $\bowtie_{k+1}$ is "=". Note that the number m of columns in A is in principle unbounded, but by Carathéodory's Theorem (Proposition 1), if a set of restrictions of the form (4.4) has a solution, then it has a "small" solution in which at most $k+1$ elements of λ are nonzero.

Given the algebraic formulation in (4.4) for a regular ŁIP-assignment as in (4.2), based on the proof by Bova and Flaminio (2010) of ŁIPSAT NP-completeness, we can assure that there exists a polynomial size witness consisting of A and λ in (4.4) for a reasonable encoding of a satisfiable ŁIP-assignment in (4.2) — that is an encoding with a size that is polynomial in k, in the number

of occurrences of operators in α_i and in the binary representation of q_i, for $i = 1, \ldots, k$. That is so because, given such a satisfiable ŁIP-assignment with encoding size η, first, there is a matrix A whose entries are rational numbers with polynomial binary representation of size at most $2\eta^2$; second, by Carathéodory's Theorem (Proposition 1), we can assume such matrix A to have at most $k+1$ rows; also, the solution λ for the linear constraints $A\lambda \bowtie q$ has polynomial size in the representations of A and q.

Note also that, since deciding the satisfiability of a ŁIP-assignment of the form $\{C(\alpha_i) = 1 \mid 1 \leq i \leq k\}$ is equivalent to deciding if the Ł_∞-SAT instance $\{\alpha_1, \ldots, \alpha_k\}$ is satisfiable, which is an NP-complete problem (Mundici, 1987), it follows that ŁIPSAT is NP-hard and so NP-complete.

Proposition 4 (Bova and Flaminio (2010)) *The problem of deciding the satisfiability of a ŁIP-assignment is NP-complete.* □

Before applying linear algebraic methods to efficiently solve ŁIPSAT, we first provide a normal form for it based on the framework of semantics modulo satisfiability.

4.2.1 A Normal Form for ŁIP-Assignments

An extended assignment may seem more expressive than regular ŁIP-assignments, but we show that no expressivity is gained by this extension. In fact, we define an *(atomic) normal form* ŁIP-assignment as a pair $\langle \Gamma, \Theta \rangle$, where Γ is a set of formulas and Θ is a set of ŁIP-assignments over propositional variables of the form

$$\Theta = \Big\{ C(P_i) = q_i \;\Big|\; q_i \in [0,1] \cap \mathbb{Q},\ P_i \in \mathbb{P},\ 1 \leq i \leq k \Big\}. \tag{4.5}$$

The formulas $\gamma \in \Gamma$ represent ŁIP-assignments of the form $C(\gamma) = 1$, that is, a set of hard constrains in the form of formulas which must be satisfied by all partial valuations in the convex combination that compose a Ł_∞-probability distribution. In terms of semantics modulo satisfiability, partial valuations $v_1, \ldots, v_m$ in (4.1) must be in $\mathbf{Val}_\Gamma^{\mathcal{P}}$; thus, a normal form ŁIP-assignment comes down to probabilistic assignments in Θ that should be in accordance with propositional theory $\text{Th}(\Gamma)$, just like normal form instances of PSAT (Section 2.3).

A normal form assignment $\langle \Gamma, \Theta \rangle$ is satisfiable if there are partial valuations $v_1, \ldots, v_m \in \mathbf{Val}_\Gamma^{\mathcal{P}}$, with $\text{Var}(\Gamma) \subseteq \mathcal{P}$ and $P_i \in \mathcal{P}$, $1 \leq i \leq k$, and there is a Ł_∞-probability distribution $\lambda = [\lambda_1, \cdots, \lambda_m]'$, such that for each assignment $C(P_i) = q_i \in \Theta$, $\sum_{j=1}^{m} \lambda_j \cdot v_j(P_i) = q_i$. The satisfiability of extended ŁIP-assignments reduces to that of normal form ones as follows.

Theorem 19 (Atomic Normal Form) *For every extended ŁIP-assignment Σ there exists a normal form ŁIP-assignment $\langle \Gamma, \Theta \rangle$ such that Σ is satisfiable iff $\langle \Gamma, \Theta \rangle$ is; the normal form assignment can be built from Σ in linear time.* □

Proof Given Σ, first transform it into Σ' in which all assignments are of the form $C(\alpha) \leq p$; for that, if Σ contains a constraint of the form $C(\alpha) \bowtie 1$, $\bowtie \in \{=, \geq\}$ (resp. $C(\alpha) = 0, C(\alpha) \leq 0$), we insert α (resp. $\neg\alpha$) in Γ and do not insert the constraint in Σ'. If $C(\alpha) = q \in \Sigma$, we insert $C(\alpha) \leq q$ and $C(\alpha) \geq q$ in Σ'. Then all assignments of the latter form are transformed into $C(\neg\alpha) \leq 1 - q$. Also, insert constraints already in the form $C(\alpha) \leq q \in \Sigma$ into Σ'. All transformation steps can be made in linear time and are such that $\Gamma \cup \Sigma'$ is satisfiable iff Σ is.

For every $C(\alpha_i) \leq q_i \in \Sigma', 0 < q_i < 1$, consider a new propositional variable P_i; insert $\alpha_i \to P_i$ in Γ and $C(P_i) = q_i$ in Θ. Clearly $\langle \Gamma, \Theta \rangle$ is in normal form and is obtained in linear time. The fact that Σ is satisfiable iff $\langle \Gamma, \Theta \rangle$ is follows from Lemma 8. ■

Example 8 Note that the formalization presented in Example 7 is already in normal form, witnessing that this format is quite a natural one to formulate ŁIP-assignments. □

4.2.2 Algebraic Methods for Normal Form ŁIP-Assignments

From now on, we assume that ŁIP-assignments are in normal form. Here we explore their algebraic structure as it allows for the interaction between a ŁIP-problem Θ and a Ł_∞-SAT instance Γ, such that solutions satisfying the normal form assignment can be seen as probabilistic solutions to Θ agreeing with propositional theory $\text{Th}(\Gamma)$.

Furthermore, to construct a convex combination of the form (4.1) we will only consider partial valuations in $\mathbf{Val}_\Gamma^\mathcal{P}$, with $\text{Var}(\Gamma) \subseteq \mathcal{P}$ and $P_i \in \mathcal{P}$, $1 \leq i \leq k$. Given a ŁIP-assignment $\langle \Gamma, \Theta = \{C(P_i) = q_i\} \rangle$, a partial valuation $v \in \mathbf{Val}^{\{P_1,\ldots,P_k\}}$ is Γ-*satisfiable* if it can be extended to a partial valuation in $\mathbf{Val}_\Gamma^\mathcal{P}$. Let us identify partial valuations $v \in \mathbf{Val}^{\{P_1,\ldots,P_k\}}$ with $(k+1)$-dimensional vectors $[v(P_1), \ldots, v(P_k), 1]'$ and let q be a $(k+1)$-dimensional vector $[q_1, \ldots, q_k, 1]'$; the following is a direct consequence of Theorem 19.

Lemma 9 *A normal form instance $\langle \Gamma, \Theta \rangle$ is satisfiable iff there is a $(k+1) \times (k+1)$-matrix A_Θ such that its entries are rational numbers with polynomial representation in the encoding of $\langle \Gamma, \Theta \rangle$, its last row is all 1's, its columns are Γ-satisfiable and $A_\Theta \lambda = q$ has a solution $\lambda \geq 0$.* □

Proof Let n be the number of formulas in Γ and let $l = n + k$. Suppose first that $\langle \Gamma, \Theta \rangle$ is satisfiable, thus the assignment admits a solution $\bar{A}\bar{\lambda} = \bar{q}$, according to (4.4); the condition $\sum \bar{\lambda}_j = 1$ is incorporated as the final row of $\bar{A}$ containing only 1's; as mentioned in Section 4.2, matrix $\bar{A}$ may be such that its entries are rational numbers with polynomial representation in the encoding of $\langle \Gamma, \Theta \rangle$. Each column $\bar{A}_j$ in matrix $\bar{A}$ corresponds to a partial valuation $v_j \in \mathbf{Val}^\mathcal{P}$ and $\bar{\lambda}$ is a Ł_∞-probability distribution over the v_j's; clearly, $\bar{\lambda}_j > 0$ implies that $v_j \in \mathbf{Val}_\Gamma^\mathcal{P}$. Let matrix $\bar{\bar{A}}$ with $k+1$ rows be obtained from $\bar{A}$ by deleting each line corresponding to a formula in Γ and deleting each column $\bar{A}_j$ such that $\bar{\lambda}_j = 0$; also, let $\bar{\bar{\lambda}}$ be obtained from $\bar{\lambda}$ by deleting each entry $\bar{\lambda}_j = 0$. Note that, by construction, $\bar{\bar{A}}\bar{\bar{\lambda}} = q$, $\bar{\bar{\lambda}} \geq 0$ and the columns in $\bar{\bar{A}}$ are Γ-satisfiable. Then, by Carathéodory's Theorem (Proposition 1) there exists a $(k+1) \times (k+1)$ matrix A_Θ, built from $\bar{\bar{A}}$ columns, and a $k+1$ dimensional vector λ such that $A_\Theta \lambda = q$ has a solution $\lambda \geq 0$.

Conversely, suppose that the desired matrix A_Θ exists, thus $A_\Theta \cdot \lambda = q$, for some $\lambda \geq 0$. Each column of A_Θ, being Γ-satisfiable, can be transformed into a column of A in (4.4) by extending it with n 1's, corresponding to the formulas in Γ. It follows easily that restrictions (4.4) have a solution and thus $\langle \Gamma, \Theta \rangle$ is satisfiable. ■

Lemma 9 leads to a linear algebraic solving method as follows. Let $V \subseteq \mathbf{Val}^{\{P_1,\ldots,P_k\}}$ be a set of partial valuations over the propositional variables in Θ that take values whose representations have a size limit based on the polynomial size of witnesses to ŁIP-assignments discussed in Section 4.2; consider a $|V|$-dimensional vector as follows.

$$c_j = \begin{cases} 0, & v_j \in V \text{ is } \Gamma\text{-satisfiable} \\ 1, & \text{otherwise} \end{cases} \tag{4.6}$$

The vector c is a Boolean "cost" associated to each partial valuation $v_j \in V$, such that the cost is 1 iff v_j is Γ-unsatisfiable. Consider a matrix A whose columns are the partial valuations in V. Now consider the linear program (4.7) which aims at minimizing that cost, weighted by the corresponding probability value λ_j.

$$\begin{array}{ll} \min & c' \cdot \lambda \\ \text{subject to} & A \cdot \lambda = q \\ & \sum \lambda_j = 1 \\ & \lambda \geq 0 \\ & A\text{'s columns are partial valuations in } V \end{array} \tag{4.7}$$

Theorem 20 *A normal form instance $\langle \Gamma, \Theta = \{C(P_i) = q_i \mid 1 \leq i \leq k\}\rangle$ is satisfiable iff the linear program (4.7) reaches a minimal solution $c'\lambda = 0$. Furthermore, if there is a solution, then there is a solution in which at most $k+1$ values of λ are not null.* □

PROOF If the linear program (4.7) reaches 0, we obtain $v_1, \ldots, v_m$ by selecting only the Γ-satisfiable columns A_j for which $\lambda_j > 0$, obtaining a convex combination satisfying Θ. So, $\langle \Gamma, \Theta\rangle$ is satisfiable. Conversely, if $\langle \Gamma, \Theta\rangle$ is satisfiable, by Lemma 9 there exists a matrix A_Θ such that all of its columns are Γ-satisfiable partial valuations in V and $A_\Theta\lambda = q$; clearly A_Θ is a submatrix of A; make $\lambda_j = 0$ when A_j is not a A_Θ column and thus $c'\lambda = 0$. Again by Lemma 9, A_Θ has at most $k+1$ columns so at most $k+1$ values of λ are not null. ■

Despite the fact that solvable linear programs of the form (4.7) always have polynomial size solutions, with respect to the size of the corresponding normal form ŁIP-assignment, the elements of linear program itself may be exponentially large, rendering the explicit representation of matrix A impractical. In the following, we present an algorithmic technique that avoids that exponential explosion.

4.3 A ŁIPSAT-Solving Algorithm

Based on the results of the previous section we are going to present an algorithm employing a linear programming technique called *column generation* (Hansen and Jaumard, 1990; Kavvadias and Papadimitriou, 1990) to obtain a decision procedure for ŁIPSAT, which we call ŁIPSAT-*solving*. This algorithm solves the potentially large linear program (4.7) without explicitly representing all columns and making use of an extended solver for $Ł_\infty$-SAT as an auxiliary procedure to generate columns.

To avoid the exponential blow of the size of matrix in (4.7), the algorithm's basic idea is to employ the simplex algorithm (see Section 2.4 for references) over a normal form ŁIP-assignment $\langle \Gamma, \Theta\rangle$, coupled with a strategy that generates cost decreasing columns without explicitly representing the full matrix A. In this process, we start with a *feasible solution*, which may contain several Γ-unsatisfiable columns. We minimize the cost function consisting of the sum of the probabilities associated to Γ-unsatisfiable columns, such that when it reaches zero, we know that the problem is satisfiable; if no cost decreasing column can be generated and the minimum achieved is bigger than zero, a negative decision is reached.

The general strategy employed here is similar to that employed in PSAT-solving (Finger and Bona, 2011; Finger and De Bona, 2015), but the column generation algorithm is considerably distinct and requires an extension of Ł_∞-SAT decision procedure.

From the input $\langle \Gamma, \Theta \rangle$, we implicitly obtain an unbounded matrix A and explicitly obtain the vector of probabilities q mentioned in (4.7). The basic idea of the simplex algorithm is to move from one feasible solution to another one with a decreasing cost. The feasible solution consists of a square matrix B, called the basis, whose columns are extracted from the unbounded matrix A. The pair $\langle B, \lambda \rangle$ consisting of the basis B and a Ł_∞-probability distribution λ is a *feasible solution* if $B \cdot \lambda = q$ and $\lambda \geq 0$. We assume that $q_{k+1} = 1$ such that the last line of B will force $\sum_G \lambda_j = 1$, where G is the set of B columns that are Γ-satisfiable. Each step of the algorithm replaces one column of the feasible solution $\langle B^{(s-1)}, \lambda^{(s-1)} \rangle$ at step $s-1$ obtaining a new feasible solution $\langle B^{(s)}, \lambda^{(s)} \rangle$. The cost vector $c^{(s)}$ is a $\{0,1\}$-vector such that $c^{(s)}_j = 1$ iff B_j is Γ-unsatisfiable. The column generation and substitution is designed such that the total cost is never increasing, that is $c^{(s)\prime} \cdot \lambda^{(s)} \leq c^{(s-1)\prime} \cdot \lambda^{(s-1)}$.

Algorithm 8 presents the top level ŁIPSAT decision procedure. Lines 1–3 present the initialization of the algorithm. We assume the vector q is in ascending order. Let D_{k+1} be a $k+1$ square matrix in which the elements on the diagonal and below are 1 and all the others are 0. At the initial step we make $B^{(0)} = D_{k+1}$, this forces $\lambda^{(0)}_1 = q_1 \geq 0$, $\lambda^{(0)}_{j+1} = q_{j+1} - q_j \geq 0, 1 \leq j \leq k$; and $c^{(0)} = [c_1, \ldots, c_{k+1}]'$, where $c_k = 0$ if column j in $B^{(0)}$ is Γ-satisfiable; otherwise $c_j = 1$. Thus, the initial state $s = 0$ is a feasible solution.

Algorithm 8 ŁIPSAT-CG: a ŁIPSAT solver via Column Generation

Input: A normal form ŁIPSAT instance $\langle \Gamma, \Theta \rangle$.
Output: **No**, if $\langle \Gamma, \Theta \rangle$ is unsatisfiable. Or a solution $\langle B, \lambda \rangle$ that minimizes (4.7).

```
1:  q := [{q_i | C(p_i) = q_i ∈ Θ, 1 ≤ i ≤ k} ∪ {1}] in ascending order;
2:  B^(0) := D_{k+1};
3:  s := 0, λ^(s) = (B^(0))^{-1} · q and c^(s) = [c_1, ..., c_{k+1}]';
4:  while c^(s)' · λ^(s) ≠ 0 do
5:          y^(s) = GenerateColumn(B^(s), Γ, c^(s));
6:          if y^(s) column generation failed then
7:                  return No;   {ŁIPSAT instance is unsatisfiable}
8:          else
9:                  B^(s+1) = merge(B^(s), b^(s))
10:                 s++, recompute λ^(s) and c^(s);
11:         end if
12: end while
13: return ⟨B^(s), λ^(s)⟩;   {ŁIPSAT instance is satisfiable}
```

Algorithm 8's main loop covers lines 5–12 which contains the column generation strategy described above. Column generation occurs at beginning of the loop (line 5) which we are going to detail below. If column generation fails the process ends with failure in line 7. Otherwise a column is removed and the generated column is inserted in a process we called *merge* at line 9. The loop ends successfully when the objective function (total cost) $c^{(s)\prime} \cdot \lambda^{(s)}$ reaches zero and the algorithm outputs a probability distribution λ and the set of Γ-satisfiable columns in B, at line 13.

The procedure *merge* is part of the simplex method which guarantees that given a $k+1$ column y and a feasible solution $\langle B, \lambda \rangle$ there always exists a column j in B such that if $B[j := y]$ is obtained from B by replacing column j with y, then there is λ' such that $\langle B[j := y], \lambda' \rangle$ is a feasible solution.

Lemma 10 *Let $\langle B, \lambda\rangle$ be a feasible solution of (4.7), such that B is non-singular, and let y be a column. Then there always exists a column j such that $\langle B[j := y], \lambda'\rangle$ is a non-singular feasible solution.* □

PROOF As $\langle B, \lambda\rangle$ is a feasible solution,

$$\sum_{i=1}^{k+1} B_i \lambda_i = q. \tag{4.8}$$

Suppose we replace column B_j by y. Due to the fact that B is not singular, there are coefficients $\beta_1, \beta_2, \ldots, \beta_{k+1}$ such that $\sum_{i=1}^{k+1} \beta_i B_i = y$ and thus

$$B_j = \frac{y}{\beta_j} - \frac{\beta_1}{\beta_j} B_1 - \cdots - \frac{\beta_{j-1}}{\beta_j} B_{j-1} - \frac{\beta_{j+1}}{\beta_j} B_{j+1} - \cdots - \frac{\beta_{k+1}}{\beta_j} B_{k+1}. \tag{4.9}$$

Substituting (4.9) for B_j in (4.8) yields:

$$\frac{\lambda_j}{\beta_j} y + \sum_{i=1}^{k+1} (\lambda_i - \frac{\beta_i}{\beta_j}\lambda_j) B_i = q. \tag{4.10}$$

Note that the coefficient of B_j in the sum is 0. We have now a new vector of coefficients λ' such that $B[j := y] \cdot \lambda' = q$. Properly choosing j guarantees $\lambda' \geq 0$. As the elements of columns B_i and y are all non-negative truth values, the set $\beta_{>0} = \{\beta_i \mid \beta_i > 0\}$ is not empty. Taking a j from the set $\{j \mid \beta_j \in \beta_{>0} \text{ and } \forall i, \beta_i \lambda_j \leq \beta_j \lambda_i\}$ implies $\lambda_i - \frac{\beta_i}{\beta_j}\lambda_j \geq 0$, for all $i \neq j$, and $\lambda_j/\beta_j \geq 0$, so $\lambda' \geq 0$. Finally, as $\beta_j > 0$ and all columns in B are linearly independent, $B[j := y]$ is non-singular. ■

Lemma 10 guarantees the existence of a column which may not be unique and further selection heuristics is necessary; in our implementation we give priority to removing columns which are associated to probability zero on a left-to-right order.

We now describe the column generation method, which takes as input the current basis B, the current cost c and the restrictions Γ; the output is a column y, if it exists, otherwise it signals **No**. The basic idea for column generation is the property of the simplex algorithm called the *reduced cost* of inserting a column y with cost c_y in the basis. The reduced cost is given by equation

$$r_y = c_y - c' B^{-1} y \tag{4.11}$$

and the simplex method guarantees that the objective function is non-increasing if $r_y \leq 0$. Furthermore the generation method is such that the column y is Γ-satisfiable, so $c_y = 0$. We thus obtain

$$c' B^{-1} y \geq 0 \tag{4.12}$$

which is an inequality on the elements of y. To force λ to be a probability distribution, we make $y_{k+1} = 1$; the remaining elements y_i are truth values of the variables in Θ, so that we are searching for solution to (4.12) such that $0 \leq y_i \leq 1, 1 \leq i \leq k$. To finally obtain column y we must extend a Ł_∞-solver that generates Γ-satisfiable partial valuations so that it also respects the linear restriction (4.12). In fact this is not an expressive extension of Ł_∞ as the McNaughton property guarantees

that (4.12) is equivalent to a formula on variables $y_1, \ldots, y_k$. In practice, we tested two ways of obtaining a joint solver for Γ and (4.12).

- Employing an SMT (SAT modulo theories) solver that can handle linear algebraic expressions such as (4.12) and the ones generated by the Ł_∞-semantics. We already used such SMT-based Ł_∞-solver in Chapter 3 for testing the implementation of the representation algorithms in Section 3.5 and for verifying neural network properties in Section 3.6.
- Using a MIP (mixed integer programming) solver that encodes Ł_∞-semantics. Equation (4.12) is simply a new linear restriction to be dealt with by the MIP solver. Ł_∞-solvers based on MIP solvers have been proposed by Hähnle (1991).

In both cases, the restrictions posed by the formulas in Γ and (4.12) are jointly handled by the semantics of the underlying solver. We have thus the following result.

Lemma 11 *There are algorithmic solutions to the problem of jointly satisfying formulas and inequalities with common variables.* □

We now deal with the problem of termination. Column generation as above guarantees that the cost is never increasing. Limit cases where infinite cost descending columns might be generated are avoidable by imposing a size restriction on the representations of such columns as established in (4.7); in practice, such restriction is achieved due to the internal precision of the extended Ł_∞-solvers. In this context, the simplex method ensures that a solvable problem always terminates if the costs always decrease and we are left with the problem of guaranteeing that the objective function does not become stationary. This is guaranteed in the implementation by a column selection strategy that respects *Bland's Rule* and also by plateau escaping strategies such as *Tabu search* (see Section 2.4 for references).

Lemma 12 *There are column selection strategies that guarantee that the Algorithm 8 always terminates.* □

Although there are no column selection heuristics that guarantee that the simplex method terminates in a polynomial number of steps, it performs very well in most practical cases and its average complexity is known to be polynomial (see Section 2.4 for references). By placing all the results above together, we can state the correctness of Algorithm 8.

Theorem 21 *Consider the output of Algorithm 8 with normal form input $\langle \Gamma, \Theta \rangle$. If the algorithm succeeds with solution $\langle B, \lambda \rangle$, then the input problem is satisfiable with distribution λ over the valuations which are columns of B. If the program outputs **No**, then the input problem is unsatisfiable. Furthermore, there are column selection strategies that guarantee termination.* □

Proof Lemma 10 guarantees that in all steps, $\langle B^{(s)}, \lambda^{(s)} \rangle$ is a feasible solution to the problem. If Algorithm 8 terminates with success, than cost zero has been reached, so by Theorem 20 the input problem is satisfiable. On the other hand, if column generation fails, it fails with a positive cost and this means there are no Γ-satisfiable columns that can reduce the cost. So, the problem in unsatisfiable. Finally, a suitable column selection strategy by Lemma 12 guarantees termination. ■

Example 9 We show the steps for the solution of Example 7. Initially, we have

$$q = \begin{bmatrix} 0.6 \\ 0.6 \\ 0.6 \\ 1 \end{bmatrix}, B^{(0)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \lambda^{(0)} = (B^{(0)})^{-1} \cdot q = \begin{bmatrix} 0.6 \\ 0 \\ 0 \\ 0.4 \end{bmatrix}, c^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}.$$

$c^{(0)}$ expresses that the first two columns of $B^{(0)}$ are Γ-satisfiable. The total cost $\text{cost}^{(0)} = c^{(0)\prime} \cdot \lambda^{(0)} = 0.4$. At this point, column $y^{(1)}$ is generated substituting $B^{(0)}$'s column 3 in the *merge* procedure:

$$y^{(1)} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}, B^{(1)} = \begin{bmatrix} 1 & 0 & \mathbf{1} & 0 \\ 1 & 1 & \mathbf{0} & 0 \\ 1 & 1 & \mathbf{1} & 0 \\ 1 & 1 & \mathbf{1} & 1 \end{bmatrix}, \lambda^{(1)} = \begin{bmatrix} 0.6 \\ 0 \\ 0 \\ 0.4 \end{bmatrix}, c^{(1)} = \begin{bmatrix} 0 \\ 0 \\ \mathbf{0} \\ 1 \end{bmatrix}.$$

$\text{cost}^{(1)} = 0.4$. Again, column generation provides $y^{(2)}$ in place of column 1:

$$y^{(2)} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}, B^{(2)} = \begin{bmatrix} \mathbf{1} & 0 & \mathit{1} & 0 \\ \mathbf{1} & 1 & \mathit{0} & 0 \\ \mathbf{0} & 1 & \mathit{1} & 0 \\ \mathbf{1} & 1 & \mathit{1} & 1 \end{bmatrix}, \lambda^{(2)} = \begin{bmatrix} 0.3 \\ 0.3 \\ 0.3 \\ 0.1 \end{bmatrix}, c^{(2)} = \begin{bmatrix} \mathbf{0} \\ 0 \\ \mathit{0} \\ 1 \end{bmatrix}.$$

$\text{cost}^{(2)} = 0.1$. Finally, column generation provides $y^{(3)}$ in place of column 4:

$$y^{(3)} = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 1 \end{bmatrix}, B^{(3)} = \begin{bmatrix} \mathit{1} & 0 & \mathit{1} & \mathbf{0.5} \\ \mathit{1} & 1 & \mathit{0} & \mathbf{0.5} \\ \mathit{0} & 1 & \mathit{1} & \mathbf{0.5} \\ \mathit{1} & 1 & \mathit{1} & \mathbf{1} \end{bmatrix}, \lambda^{(3)} = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.2 \\ 0.4 \end{bmatrix}, c^{(3)} = \begin{bmatrix} \mathit{0} \\ 0 \\ \mathit{0} \\ \mathbf{0} \end{bmatrix}.$$

$\text{cost}^{(3)} = 0$, so that the problem is satisfiable with solution $\langle B^{(3)}, \lambda^{(3)} \rangle$. □

4.4 Implementation and Results

We have developed implementations of solvers for the Ł$_\infty$-SAT and ŁIPSAT problems. In this section, we present the empirical results on the search for the qualitative behavior of phase transition. The source code for all experiments under license GPLv3 is publicly available.[3]

A decision problem displays a *phase transition* when there is an ordering of classes of instances that presents a transition from predominantly satisfiable instances (answer **Yes**) to predominantly unsatisfiable instances (answer **No**), which is called a *first-order phase transition*. Furthermore, the decision problem displays a peak in average execution time around the middle of that transition in which fifty percent of answers are **Yes** and fifty percent of answers are **No**, which is called a *second-order phase transition*, following the terminology of statistical mechanics (Cheeseman *et al.*, 1991).

It is conjectured that there is a (second-order) phase transition for every NP-complete decision

[3] http://lipsat.sourceforge.net

problem (Cheeseman *et al.*, 1991). Empirical phase transition behavior are well established for classical SAT (Gent and Walsh, 1994) and PSAT (Finger and Bona, 2011), among many others. In fact, the empirical verification of phase transition for solvers of an NP-complete problem can be perceived as a quality test for its implementation.

4.4.1 Phase Transition for Ł$_\infty$-Solvers

In a classical setting one usually employs 3-SAT format[4] to obtain a phase transition diagram. The randomly generated formulas are clauses with three literals each, the number of symbols n is fixed and the rate between the number n of clauses and the rate $\frac{m}{n}$ is used as the control parameter, where m is the number of clauses. In classical 3-SAT, the shape of the curve and the phase transition point is maintained when n is changed. Unfortunately, there is no clausal normal form for formulas in Ł$_\infty$. So, we employ instead a set of formulas which are used by Bofill *et al.* (2015) consisting of

$$l_1 \oplus l_2 \oplus l_3 \tag{4.13}$$

$$\neg(l_4 \oplus l_5) \oplus l_6 \tag{4.14}$$

where l_i are literals (negated or non-negated atoms). The generation of the formulas is parameterized by the number n of propositional variables and the number m of formulas, which define the class of randomly generated formulas. Following Bofill *et al.* (2015), formulas are generated as follows: 70% of formulas are of format in (4.13) and 30% of the format in (4.14). Each literal is randomly chosen from the n possible symbols with equal probability, then there is a 50% chance of being a positive or negative literal.

Two implementations were developed using publicly available open source software:

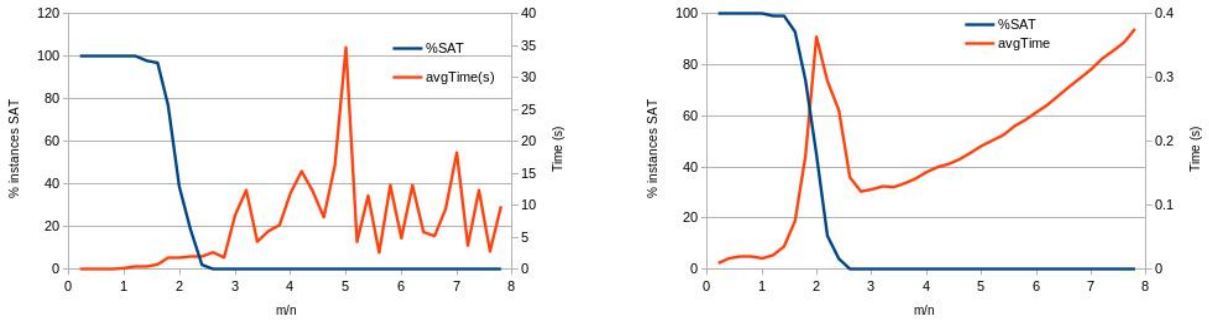
- a C++-implementation using the C++ interface to the Yices SMT solver (Dutertre, 2014);
- a C++-implementation using the C++ interface to the SCIP MIP solver (Achterberg, 2009).

For each implementation, the experiment proceeds as follows: with a fixed $n = 100$ we varied the value of m such that the rate $\frac{m}{n}$ varies from 0.2 to 8 in 0.2 steps. For each pair $\langle n, m\rangle$ we construct a set of 100 randomly generated formulas as described above. And for each set we compute the percentage of Ł$_\infty$-satisfiable formulas and the average decision time (user time).

All the experiments in this section were run on a UNIX machine with a i7-6900K CPU @ 3.20GHz with 16 processors. The results of the experiments using two Ł$_\infty$-solvers are shown in Figure 4.1. In Figure 4.1a we see the results of an SMT(LA) Ł$_\infty$-solver using YICES which presents a first-order phase transition from satisfiable to unsatisfiable instances with a middle point occurring at rate $\frac{m}{n} \approx 2$; however, the average decision time peak occurs at $\frac{m}{n} \approx 5$, unlike what is expected. Furthermore, the peak time for solving a Ł$_\infty$-SAT problem is about 35 seconds. This unexpected behavior may be credited to the fact that YICES converts internally all floating point numbers to pair of integers, which impacts the efficiency of problems whose formulation involves a lot of floating point numbers as is the case of Ł$_\infty$-SAT.

Figure 4.1b presents an Ł$_\infty$-solver using MIP solver SCIP, in which we can see a phase transition from satisfiable to unsatisfiable instances also at $\frac{m}{n} \approx 2$, with an average time peak also around

[4] 3-SAT is the usual name for the CPL-SAT restricted to CNF CPL-formulas whose clauses have at most three literals.



(a) *Based on SMT(LA) using YICES* **(b)** *Based on MIP solver using SCIP*

Figure 4.1: *Ł$_\infty$-solvers performance, randomly gen. instances:* $n = 100$, $m = 20$ *to* 780

$\frac{m}{n} \approx 2$, as expected. Furthermore, the peak time is 0.35 seconds, two orders of magnitude more efficient than the YICES solver. Observing the average time, we note an always increasing right tail, which can be credited to the fact that MIP solvers are not implemented with a "fail early" strategy commonly used in logic-based solvers, which normally employ what is called restriction learning strategies; furthermore, the size of the matrices used by the MIP solver increases with m. Another possibility to explain such a behavior is the fact that the choice of the family of formulas may be inappropriate, however no such increasing tail was observed in the SMT-based method, which reinforces the hypothesis that this behavior is due to the MIP solver. Due to its superior efficiency we only use the SCIP solver as an auxiliary procedure for the ŁIPSAT solver described next.

4.4.2 Phase Transition for ŁIPSAT

The input for the ŁIPSAT solver is a normal form $\langle \Gamma, \Theta \rangle$. We developed a C++-implementation for Algorithm 8 using the C++ interface of the SoPlex linear algebra solver which is part of the SCIP suite of optimizers. We used the Ł$_\infty$-solver based on SCIP MIP.

The experiments were obtained as follows. The input set of formulas Γ was generated with a fixed number of symbols n and a varying number of clauses of format (4.13) and (4.14) as described above. The probabilistic Θ-restrictions of the form $\{C(y_i) = q_i \mid i \leq i \leq k\}$ were generated fixing $k \leq n$ and randomly choosing the probabilities q_i uniformly over the interval $[0, 1]$.

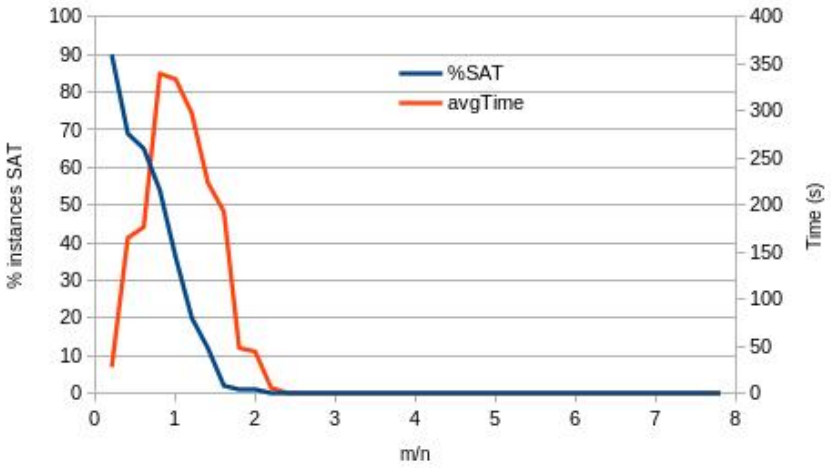


Figure 4.2: *Phase transition for ŁIPSAT solver:* $k = 20$, $n = 100$ *and* $m = 20$ *to* 780

The results of the experiment can be seen in Figure 4.2. We clearly see a second-order phase transition with a peak average time execution that overlaps the decreasing part of the percentage satisfiable (%SAT) curve. Note that no increasing tail is observed, so that the "fail early" mechanism is achieved in the combination of logic and linear algebra. The peak is near but does not coincide with the fifty percent point of the first order phase transition which may be credited to the increasing shape of the right tail in the $Ł_\infty$-solver presented in Figure 4.2. Also, there is a left shift of the phase transition point $\frac{m}{n} \approx 1$, similar to the shift of PSAT phase transition point with respect to CPL-SAT (Finger and De Bona, 2015). Overall the phase transition format can be considered satisfactory.

The results in this chapter have appeared in the following publications.

- Finger and Preto (2020) Marcelo Finger and Sandro Preto. Probably partially true: Satisfiability for Łukasiewicz infinitely-valued probabilistic logic and related topics. *Journal of Automated Reasoning*, 64(7):1269-1286. ISSN 1573-0670. doi: 10.1007/s10817-020-09558-9. URL http://doi.org/10.1007/s10817-020-09558-9.

- Finger and Preto (2018) Marcelo Finger and Sandro Preto. Probably half true: Probabilistic satisfiability over Łukasiewicz infinitely-valued logic. In Didier Galmiche, Stephan Schulz and Roberto Sebastiani, editors, *Automated Reasoning. IJCAR 2018*, volume 10900 of *Lecture Notes in Computer Science*, pages 194-210, Cham. Springer International Publishing. ISBN 978-3-319-94205-6.

Chapter 5

Probabilistic Constraints on Nash Equilibria

In this chapter, we study a scenario called observable game for which: there are players who will perform actions among the ones available for each of them; it is known that the combined actions of these players will have the property of being a Nash equilibrium (Nash, 1951); and there is an external observer who is aware of that, however is unsure about which action each player will choose. Therefore, such external observer assigns subjective probability constraints to actions representing his degree of confidence that such actions will be performed. The following problems are considered according to the concepts of pure and mixed Nash equilibrium.

The Coherence Problem Given an observable game — a game together with a set of probabilistic constraints on its actions —, decide if it is coherent; that is, decide if there exists an actual probability distribution on the game equilibria that corresponds to those probabilistic constraints.

The Extension (Inference) Problem Given a coherent observable game with probabilistic constraints on some of the players' actions, compute upper and lower bounds on the probabilities of some other action that preserves coherence.

The frameworks of semantics modulo satisfiability coupled with probabilistic assignments to logical formulas yield the means to manage the scenario of an observable game. First, in a setting of propositional logic where valuations encode the player's strategies for the game, we determine axioms for a propositional theory that are only satisfiable by valuations encoding equilibria. Thus, we identify the observable game with a PSAT or a ŁIPSAT instance — depending on whether only pure equilibria or also mixed equilibria are allowed — which states probabilistic assignments that intend to agree with the aforementioned propositional theory.

For the Coherence Problems, we provide complexity classification and algorithms through polynomial reductions from them to the well-studied decision problems associated to probability theories grounded on propositional logics. In turn, we study the Extension Problems via reductions to the Coherence Problems.

We first analyze the scenario modeled by the framework studied in this chapter in Section 5.1. In Section 5.2, we introduce the formal notions of game, observable game and the Coherence and Extension problems and also introduce pure equilibrium and study its computation; Section 5.4 has

analogous content for the mixed equilibrium setting. In Section 5.3, we study and propose algorithms for the Coherence and Extension problems in the pure equilibrium setting; in Section 5.5, we study these problems in the mixed equilibrium setting. Finally, in Section 5.6, we analyze the implications of our results for the phenomena modeled by observable games.

In this chapter, when we talk about formulas, valuations, partial valuations and **Val**, we refer to CPL-formulas, CPL-valuations, CPL-partial valuations and CPL-**Val** in Sections 5.2 and 5.3; and to Ł$_\infty$-formulas, Ł$_\infty$-valuations, Ł$_\infty$-partial valuations and Ł$_\infty$-**Val** in Sections 5.4 and 5.5.

5.1 Motivation on Observable Games

In game theory, a Nash equilibrium represents a situation in which each player's strategy is a best response to other players' strategies; thus no player can obtain gains by changing alone his own strategy. Nash proved that every n-player, finite, non-cooperative game has a mixed equilibrium point (Nash, 1951, 1950a,b); however, more than one equilibrium may exist and the number of equilibria can be even exponentially large over some game parameters.

For an observer knowing that an equilibrium is to be reached, there is an a priori uncertainty before an instance of the game starts, concerning the exact kind of equilibrium to be reached and also in knowing the players' actions in that instance. In such a scenario, which we call an observable game, it is most natural to describe the outcome in terms of subjective probabilities assigned to actions, in which one presupposes a probability distribution over the set of all possible equilibria.

Unfortunately, not every assignment on action probabilities by an observer finds correspondence to an actual probability distribution on possible equilibria of a given game; in fact, some actions may always co-occur at equilibrium, so constraining their probabilities to distinct values does not correspond to any underlying distribution on equilibria. In case the observer assigns a set of probabilistic constraints on actions that correspond to a probability distribution on equilibria, we say the observable game is coherent.

Lack of coherence can have important consequences which are better seen in a betting scenario where an observer knows the configuration of a game before one of its instances is played and also knows that this game reaches an equilibrium. The observer wants to place bets on the occurrence of actions and an incoherent set of probabilities may lead to sure loss. So detecting and avoiding such a disastrous assignment of probabilities may have considerable cost to the observer. This betting scenario corresponds to de Finetti's interpretation of subjective probabilities (de Finetti, 1931, 1937, 2017) in which incoherent probabilities have a one-to-one correspondence to sure loss.

An actual scenario of this kind may be seen in the pricing strategy of oligopolistic markets. Assume that only a few companies dominate a beer market. They price their products from time to time in light of competition aiming to conquer the largest market share and make the most profit. Among the mechanisms of sale strategies there are price promotions (short-term price reductions), thus the price portfolio of a company in some period is not of public knowledge in advance. However, it is very reasonable to assume that the profits of the companies in the oligopoly reach an equilibrium during the sales period under consideration. Oblivious to the oligopoly competition, it is of great interest to a local brewer to predict the price portfolios of the big companies based on his experience in observing their competition and pricing strategies; such prediction might help the local brewer to set up his own pricing strategy and even his production process, which takes place in a small

and more limited industry. This information may be crucial, for example, for deciding to limit the production of some specific beers that cannot be competitive with the oligopoly price portfolios of that period and focus on the production of some other beers with a more targeted niche, or even launch new non-beer products. In this scenario, the big companies, their price portfolios and their profits (which may be inferred from their price portfolios), are respectively the players, their actions and their utilities in a game; the local producer with predictions about the oligopolistic market is the observer with subjective probabilities over the player's actions.

Observable games formalize such scenario with a market in equilibrium and an external agent who has some idea about that equilibrium but is uncertain on the probabilistic distribution on the possible equilibria and therefore on the players' actions. Of course, there may be aspects left out as it is expected from any theoretical idealization of the real world.

In order to better understand the combination of uncertainty and game equilibria, we initially concentrate on uncertainty over pure equilibria, a restricted form of mixed equilibria in which each player chooses as strategy a unique action and whose existence is not even guaranteed. That is, the observer knows a priori that a pure equilibrium is to be reached for a given game, but does not know exactly which actions will be performed at equilibrium. We later consider uncertainty in mixed equilibria, a doubly uncertain situation, that combines uncertainty on the actions to be played in a specific instance of a game with the also probabilistic notion of mixed strategy.

5.2 Observable Games and Coherence

Define a *game* as a quadruple $G = \langle P, N, A, u\rangle$, where $P = \{1, \ldots, n\}$ lists the n *players* in the game, $N = \langle N_1, \ldots, N_n\rangle$ is a sequence of player *neighborhoods* in which $N_i \subseteq P \setminus \{i\}$ is the set of player i *neighbors*, $A = A_1 \times \cdots \times A_n$ is a set of *action profiles* in which each A_i is the set of all possible *actions* for player i and $u = \langle u_1, \ldots, u_n\rangle$ is a sequence of *utility functions* in which $u_i : A \to \mathbb{Q}$ is the utility function for player i. Assume that $A_i \cap A_j = \varnothing$ for player $i \neq j$ and that $u_i(a_1, \ldots, a'_j, \ldots, a_n) = u_i(a_1, \ldots, a''_j, \ldots, a_n)$, for $j \notin N_i \cup \{i\}$.

An action profile $e = \langle a_1, \ldots, a_i, \ldots, a_n\rangle$ is a *pure (Nash) equilibrium* if, for every player i, $u_i(e) \geq u_i(a_1, \ldots, a'_i, \ldots, a_n)$ for every $a'_i \in A_i$. A game G may have zero or more pure equilibria.[1] We write $a_i \in e$ to express that a_i is the ith component of e.

By an *observable game* we mean a pair $\mathcal{G} = \langle G, \Pi\rangle$, where G is a game and Π is a set of *probabilistic constraints on equilibria (PCE)*, that is a set of probability assignments on actions limiting the probabilities of some actions occurring in an equilibrium, which represents the observer's ignorance on what equilibrium will be reached; we assume it has the following format:

$$\Pi = \Big\{ P(\alpha_k) \;\bowtie_k\; p_k \;\Big|\; \bowtie_k \in \{\leq, \geq, =\},\ 1 \leq k \leq K \Big\}, \tag{5.1}$$

where α_k are actions and p_k are values in $[0, 1] \cap \mathbb{Q}$; we only consider rational probabilities because we are concerned with computational problems.

As observable games deal with the scenario where an equilibrium is to be reached but its action profile is unknown, we assign probabilities to equilibria:[2] let $E_G = \{e_1, \ldots, e_M\}$ be the

[1] Only mixed equilibria are guaranteed to exist, not pure ones; but every pure equilibrium is also a mixed equilibrium.

[2] This probability function over equilibria should *not* be confused with probabilities in mixed strategies.

set of all equilibria associated with game G; we consider a probability function over G-equilibria $P : E_G \to [0,1] \cap \mathbb{Q}$, such that $P(e_i) \geq 0$ and $\sum_{e_i \in E_G} P(e_i) = 1$. We define the probability $P(a_i)$ that $a_i \in e \in E_G$ is executed in a game G as

$$P(a_i) = \sum_{j \mid a_i \in e_j} P(e_j).$$

Given a game G and an equilibrium probability function P, it is possible to compute the probability of any action; however we face two problems. First, the number of equilibria may be exponentially large in the numbers of players and of actions allowed for players. Second, we may not know the equilibrium probability function P. Instead we are presented with an observable game $\mathcal{G} = \langle G, \Pi \rangle$, where G is a game and Π is a set of PCE and we are asked to decide the existence of an underlying probability function P that satisfies Π; and, in case one exists, we want to compute the range of probabilities for an unconstrained action a_i. The former problem is called the *Probabilistic Coherence Problem* and the second one is the *Probabilistic Extension Problem*.

Definition 7 (PCE Coherence Problem) Given an observable game $\mathcal{G} = \langle G, \Pi \rangle$, PCE-Coherence consists of deciding if it is *coherent*, that is if there exists a probability function over the set of G-equilibria that satisfies all constraints in Π. PCE-Coherence rejects the instance if it is not coherent or if there exists no equilibrium in G. □

Definition 8 (PCE Extension Problem) Let $\mathcal{G}$ be a coherent observable game. Given an action $a_i \in A_i$, PCE-Extension consists in finding probability functions $\underline{P}$ and $\overline{P}$ that satisfy Π such that $\underline{P}(a_i)$ is minimal and $\overline{P}(a_i)$ is maximal. □

Example 10 Suppose we have a game between Alice and Bob in which Alice has three possible actions a^1, a^2 and a^3 and Bob also has three possible actions b^1, b^2 and b^3, such that the joint utilities are given by Table 5.1. This game has three pure Nash equilibria: $\langle a^1, b^1 \rangle$, $\langle a^2, b^3 \rangle$ and $\langle a^3, b^3 \rangle$, which are stressed in bold. Suppose the game will reach a pure equilibrium state, in which

	b^1	b^2	b^3
a^1	$\mathbf{2, 2}$	$1, 1$	$1, 0$
a^2	$1, 2$	$5, 4$	$\mathbf{1, 5}$
a^3	$0, 1$	$2, 3$	$\mathbf{1, 3}$

Table 5.1: *Utility functions for Alice and Bob*

case Bob and Alice will have chosen to play a single action; we now want to see through an external observer's eyes who does not know which equilibrium will be reached, however gives to the action a^2 the probability of $\frac{1}{3}$. Is this restriction feasible (coherent)? And if it is, what is the lower bound on the probability of Bob playing b^3 this observer should assign in order to remain coherent? Can it be, say, $\frac{1}{4}$?

Let us formalize such situation by $\mathcal{G}_1 = \langle G_1, \Pi_1 \rangle$, where $G_1 = \langle P, N, A, u \rangle$, in which $P = \{a, b\}$, $N_i = P \setminus \{i\}$, $A_a = \{a^1, a^2, a^3\}$, $A_b = \{b^1, b^2, b^3\}$ and u is given by Table 5.1. In Π_1, we consider the action a^2 occurring in an equilibrium with constraint $P(a^2) = \frac{1}{3}$. This constraint is coherent and it implies that the probability of b^3 is at least $\frac{1}{3}$. So if we consider Π_1 with joint constraints $P(a^2) = \frac{1}{3}$ and $P(b^3) = \frac{1}{4}$, $\mathcal{G}_1$ is incoherent. □

This framework where probabilities are assigned to pure equilibria is very similar to another concept of equilibrium: the *correlated equilibrium* (Aumann, 1974). A correlated equilibrium in a game $G = \langle P, N, A, u\rangle$ is a probability distribution over the set of action profiles A that satisfies a specific equilibrium property. Despite the similarity, these are distinct objects: while our framework models the uncertainty about which Nash equilibrium will be reached in a game (by a probability distribution over $E_G \subseteq A$), the distribution in a correlated equilibrium is the very concept of equilibrium and is defined over all possible action profiles (not necessarily Nash equilibria).

In a deeper comparison, for both computing a correlated equilibrium and deciding on the coherence of an observable game, it is necessary to guarantee that a probability distribution on action profiles satisfies some linear inequalities that model the equilibrium property, in the case of correlated equilibrium, and that represents the probabilistic constraints, in the case of coherence. However, while the correlated equilibrium inequalities may be directly derived from the given game (Papadimitriou and Roughgarden, 2008), in order to write the coherence inequalities, it is necessary to compute the Nash equilibria of the game, since the distribution in question is over such equilibria. This difference should explain the discrepancy in complexity between the problem of computing a correlated equilibrium, which is polynomial (Papadimitriou and Roughgarden, 2008), and that of computing a distribution over E_G satisfying a set of PCE, which is nondeterministic polynomial; indeed, the proof we provide for the NP-completeness of PCE-Coherence (concerning only pure equilibria) depends on the NP-completeness of computing pure equilibria.

5.2.1 Classes of Games

We may find in the literature several ways to represent games and this issue is directly related to the configuration of the instances for our problems and, thus, to its complexity classification. We focus on classes of games whose sizes are restricted and which possess equilibrium finding algorithms whose computation complexity is also restricted; we limit our attention to what we call *GNP-classes*, in which the representation of the game takes polynomial space in the numbers n of players and s of maximum actions allowed for each player and the computation of each of the pure equilibrium profiles may be made in nondeterministic polynomial time, also in terms of n and s. Due to the time complexity restriction, the problem of deciding the existence of equilibria in a given GNP-class has complexity in NP.

A natural way to represent games is by means of the *standard normal form game* where the neighborhood of each player is $N_i = P \setminus \{i\}$, for all $i \in P$, and it is instantiated by explicitly giving its utility functions in a table with an entry for each action profile $a \in A$ containing a list with player utilities $u_i(a)$, for all $i \in P$, as in Example 10.

It is an easy task to compute a pure Nash equilibrium of a game when its player utility functions are given extensively, as in standard normal form. In that case, we just need to check, for each action profile $e = \langle a_1, \ldots, a_i, \ldots, a_n\rangle$, whether it is a pure Nash equilibrium by comparing $u_i(e)$ with $u_i(a_1, \ldots, a'_i, \ldots, a_n)$, for all $i \in P$ and $a'_i \in A_i$. For each of the $|A|$ action profiles, $\sum_{i\in N} |A_i|$ comparisons will be needed. As the instance of the game is assumed to comprehend the utility function values for all players, the computation can be done in polynomial time in the size of the instance. However, in this explicit and complete format, the instance is exponential in the number n of players, for if each player has exactly s actions, each utility function has s^n values and the game instance has ns^n values to represent all utility functions. Therefore, a class of standard normal form

	a^1	a^2	a^3
c^1	10	10	5
c^2	5	10	0
c^3	5	0	10

	b^1	b^2	b^3
a^1	10	5	0
a^2	10	10	5
a^3	5	0	10

	c^1	c^2	c^3
b^1	10	5	0
b^2	10	10	5
b^3	5	0	10

Table 5.2: *Utility functions for players a, b and c, respectively*

games fails to be a GNP-class since, despite equilibria being computable in polynomial time, the utility function requires exponential space to be explicitly represented.

More compact game representations, along with the complexity issues on deciding the existence of pure equilibria on them may be found in Gottlob *et al.* (2005). A *graphical normal form game* is such that utility functions are extensively given in separate tables, for each player i, with an entry for each element in $\times_{j \in N_i \cup \{i\}} A_j$ containing a correspondent utility value $u_i(a)$, where it is enough to consider only the entries in a with indices in $N_i \cup \{i\}$, since, as defined earlier, $u_i(a_1, \ldots, a'_j, \ldots, a_n) = u_i(a_1, \ldots, a''_j, \ldots, a_n)$ for $j \notin N_i \cup \{i\}$. Graphical normal form games can be turned into a compact representation by imposing the *bounded neighborhood* property: let k be a constant, we say that a game has k-bounded neighborhood if $|N_i| \leq k$, for all $i \in P$.

Example 11 Let $G_2 = \langle P, N, A, u \rangle$ be a game with $P = \{a, b, c\}$, $A_a = \{a^1, a^2, a^3\}$, $A_b = \{b^1, b^2, b^3\}$, $A_c = \{c^1, c^2, c^3\}$ and utility functions given by Table 5.2, from which on can infer the set N. G_2 is a game in graphical normal form with k-bounded neighborhood for $k \geq 1$, where for each player utility, only the previous player's action matters. As $k < n - 1$, G_2 has a more compact representation than it would have in standard normal form. Note that this instance of graphical normal form game has 27 utility values explicitly represented and the same game in standard normal form would need 81 utility values.

It was shown by Gottlob *et al.* (2005) that the problem of deciding whether a graphical normal form game has pure Nash equilibria is NP-complete and, by Fischer *et al.* (2006), that NP-hardness holds even when the game has 2-bounded neighborhood, where each player can choose from only 2 possible actions and the utility functions range among 2 values. It is trivial to establish a nondeterministic polynomial algorithm for computing pure Nash equilibria on these games by guessing and then verifying it.

Thus, we establish GNP-classes that contain the games in graphical normal form with k-bounded neighborhood and at most s actions allowed to each player, for fixed $k \geq 2$ and $s \geq 2$; let GNP_k^s represent these classes. Since it is needed at most ns^k values to represent the utility functions, the representation of the games uses polynomial space in the number n of players. Also, $\text{GNP}_k = \bigcup_{s \in \mathbb{N}} \text{GNP}_k^s$ are GNP-classes where the representation of the games uses polynomial space in the number n of players and the number s of maximum actions allowed.[3] Note that deciding the existence of pure equilibria in GNP_k^s and GNP_k are NP-complete problems; we refer to the GNP-classes with this property as NP-complete GNP-classes.

5.2.2 Computing Pure Nash Equilibria via CPL-SAT

The Cook-Levin Theorem (Cook, 1971) guarantees that there exists a polynomial reduction from the problem of computing pure equilibria on GNP-classes to CPL-SAT. Let us show such a reduction.

[3] It is also necessary that the representation sizes of the utility function values be bounded by a polynomial in n and s.

Given a game G with $P = \{1, \ldots, n\}$ and $A_i = \{a_i^1, \ldots, a_i^{s_i}\}$, for $i \in P$, we build a set of clauses Φ_G over propositional variables X_i^j meaning that player i chose action a_i^j. Let k be the maximal $|N_i|$ and s be the maximal $|A_i|$, $s_i \leq s$. The set Φ_G is built as follows:

(a) For each player i, add a clause $\bigvee_{j=1,\ldots,s_i} X_i^j$, representing that each player chooses one action. This set of clauses is built in time $O(ns)$.

(b) For each player i and pair a_i^p, a_i^q, with $p \neq q$, add a clause $\neg X_i^p \vee \neg X_i^q$, representing that each player chooses only one action. This set of clauses is built in time $O\left(n\binom{s}{2}\right)$.

(c) For each player i and $a = \langle a_1^{q_1}, \ldots, a_{i-1}^{q_{i-1}}, a_{i+1}^{q_{i+1}}, \ldots, a_n^{q_n}\rangle$, add the clause $\bigvee_{j\in N_i} \neg X_j^{q_j} \vee \bigvee_{r\in R} X_i^r$, where R is the set of indexes r such that $u_i(a_1^{q_1}, \ldots, a_{i-1}^{q_{i-1}}, a_i^r, a_{i+1}^{q_{i+1}}, \ldots, a_n^{q_n}) \geq u_i(a_1^{q_1}, \ldots, a_{i-1}^{q_{i-1}}, a_i', a_{i+1}^{q_{i+1}}, \ldots, a_n^{q_n})$, for all $a_i' \in A_i$, representing each player chooses one of the best responses depending on his neighborhood choices; there may be more than one best response all of which have the same utility. This set of clauses is built in time $O(ns^k)$.

For games in GNP_k^s, Φ_G is built in linear time in n and for games in GNP_k, it is built in polynomial time in n and s. A nondeterministic polynomial algorithm for computing pure Nash equilibria consists of the aforementioned reduction from a game G to its corresponding set of clauses Φ_G, with $\mathrm{Var}(\Phi_G) = \mathcal{P} \subseteq \mathbb{P}$, followed by a nondeterministic algorithm computing a partial valuation $v \in \mathbf{Val}^{\mathcal{P}}$ satisfying Φ_G, that is $v \in \mathbf{Val}^{\mathcal{P}}_{\Phi_G}$. The partial valuations $v \in \mathbf{Val}^{\mathcal{P}}_{\Phi_G}$ naturally encode action profiles that are pure equilibria and, conversely, any pure equilibrium e corresponds to a partial valuation $v_e \in \mathbf{Val}^{\mathcal{P}}_{\Phi_G}$.

Example 12 For the game G_1 in Example 10, the set of formulas Φ_{G_1} contains the variables X_a^1, X_a^2, X_a^3, X_b^1, X_b^2, X_b^3 and the following clauses.

(a) $X_a^1 \vee X_a^2 \vee X_a^3$, $X_b^1 \vee X_b^2 \vee X_b^3$.

(b) $\neg X_a^1 \vee \neg X_a^2$, $\neg X_a^1 \vee \neg X_a^3$, $\neg X_a^2 \vee \neg X_a^3$, $\neg X_b^1 \vee \neg X_b^2$, $\neg X_b^1 \vee \neg X_b^3$, $\neg X_b^2 \vee \neg X_b^3$.

(c) $\neg X_b^1 \vee X_a^1$, $\neg X_b^2 \vee X_a^2$, $\neg X_b^3 \vee X_a^1 \vee X_a^2 \vee X_a^3$, $\neg X_a^1 \vee X_b^1$, $\neg X_a^2 \vee X_b^3$, $\neg X_a^3 \vee X_b^3$. □

Example 13 For the game G_2 in Example 11, the set of formulas Φ_{G_2} contains the variables X_a^1, X_a^2, X_a^3, X_b^1, X_b^2, X_b^3, X_c^1, X_c^2, X_c^3 and the following clauses.

(a) $X_a^1 \vee X_a^2 \vee X_a^3$, $X_b^1 \vee X_b^2 \vee X_b^3$, $X_c^1 \vee X_c^2 \vee X_c^3$.

(b) $\neg X_a^1 \vee \neg X_a^2$, $\neg X_a^1 \vee \neg X_a^3$, $\neg X_a^2 \vee \neg X_a^3$, $\neg X_b^1 \vee \neg X_b^2$, $\neg X_b^1 \vee \neg X_b^3$, $\neg X_b^2 \vee \neg X_b^3$, $\neg X_c^1 \vee \neg X_c^2$, $\neg X_c^1 \vee \neg X_c^3$, $\neg X_c^2 \vee \neg X_c^3$.

(c) $\neg X_c^1 \vee X_a^1 \vee X_a^2$, $\neg X_c^2 \vee X_a^2$, $\neg X_c^3 \vee X_a^3$, $\neg X_a^1 \vee X_b^1$, $\neg X_a^2 \vee X_b^1 \vee X_b^2$, $\neg X_a^3 \vee X_b^3$, $\neg X_b^1 \vee X_c^1$, $\neg X_b^2 \vee X_c^1 \vee X_c^2$, $\neg X_b^3 \vee X_c^3$. □

5.3 From PCE-Coherence to PSAT

Let us first formulate PCE-Coherence in linear algebraic terms. Let $\mathcal{G} = \langle G, \Pi \rangle$ be an observable game where G is a game with M pure Nash equilibria and $\Pi = \{P(\alpha_i) \bowtie_i p_i \mid 1 \leq i \leq K\}$ is a set of PCE; consider a $K \times M$ matrix $A = [a_{ij}]$ such that $a_{ij} = 1$, if $\alpha_i \in e$, where e is the j-th pure Nash equilibrium of G, and $a_{ij} = 0$, otherwise. Then, PCE-Coherence is to decide if there is a probability vector π of dimension M that obeys:

$$\begin{array}{rcl} A\pi & \bowtie & p \\ \sum \pi_j & = & 1 \\ \pi & \geq & 0 \end{array} \tag{5.2}$$

Since it is not mandatory for the PCE-Coherence instance to attach a constraint to each action, matrix A may have fewer lines than the number of actions involved. As done sometimes before, we may join the first two conditions in (5.2) in just one matrix A. The next results establish computational complexity for PCE-Coherence.

Theorem 22 PCE-Coherence *over a GNP-class is a problem in NP.* □

Proof Suppose the observable game $\mathcal{G} = \langle G, \Pi \rangle$ is coherent and $|\Pi| = K$. Therefore there exists a probability distribution $\bar{\pi}$ over the set of all possible pure Nash equilibria that satisfy Π. By the Carathéodory's Theorem (Proposition 1) there is a probability distribution π assigning nonzero probabilities to at most $K + 1$ equilibria. These equilibria are polynomially bounded in size since G is member of a GNP-class. Since π is also polynomially bounded, there is a polynomial witness attesting that Π is satisfiable. Therefore, $\mathcal{G}$ is coherent and PCE-Coherence is in NP. ■

Theorem 23 PCE-Coherence *over an NP-complete GNP-class is NP-complete.* □

Proof Membership in NP follows from Theorem 22. For NP-hardness, let us reduce the problem of deciding the existence of pure Nash equilibria for games in the NP-complete GNP-class at hand to PCE-Coherence over this same class. Given a game $G = \langle P, N, A, u \rangle$, we consider the instance of observable game $\mathcal{G} = \langle G, \{P(a_i) \geq 0\} \rangle$, for some arbitrary $a_i \in A_i$, for $i \in P$. The reduction from G to $\mathcal{G}$ may be computed in linear time; and $\mathcal{G}$ is coherent if, and only if, G has a pure Nash equilibrium. We have shown that PCE-Coherence is NP-hard. ■

Corollary 3 PCE-Coherence *over* GNP_k^s *and* GNP_k *are NP-complete.* □

The algebraic formulation of PCE-Coherence in (5.2) resembles the one of PSAT in Section 2.3 and, indeed, motivates the following reduction from the former to the latter problem. Let $\mathcal{G} = \langle G, \Pi \rangle$ be an observable game such that G is member of a GNP-class and $\Pi = \{P(\alpha_i) \bowtie_i p_i \mid 1 \leq i \leq K\}$ is a set of PCE. Let $\mathcal{P}$ be a set of propositional variables in one-to-one correspondence to all possible actions in G and denote by $X_i \in \mathcal{P}$ the propositional variable associated to each action α_i appearing in Π. The PSAT instance Σ_G we construct is such that $\Pi_{\mathcal{P}} = \{P(X_i) \bowtie_i p_i \mid 1 \leq i \leq K\} \subseteq \Sigma_G$.

Semantics modulo satisfiability comes into play to ensure that the PSAT instance forces a probability distribution that only assigns nonzero probability over valuations that represent Nash

equilibria. Since G is in a GNP-class, by the reduction in Section 5.2.2, there is a set of formulas Φ_G, with $\mathrm{Var}(\Phi_G) = \mathcal{P}$, such that if we have a partial valuation $v \in \mathbf{Val}^{\mathcal{P}}_{\Phi_G}$, then $\{X \in \mathcal{P} \mid v(X) = 1\}$ is a set of propositional variables representing actions which jointly played are in equilibrium. Thus, we claim that the desired PSAT instance is $\Sigma_{\mathcal{G}} = \Pi_{\mathcal{P}} \cup \{P(\varphi) = 1 \mid \varphi \in \Phi_G\}$. If all symbols $\bowtie_i$ in $\Pi_{\mathcal{P}}$ are the equality symbol, the PSAT instance may be put in the equivalent atomic normal form $\langle \Phi_G, \Pi_{\mathcal{P}} \rangle$.

Theorem 24 *Let $\mathcal{G} = \langle G, \Pi \rangle$ be an observable game, where G is a member of a GNP-class and Π is a set of PCE, and $\Sigma_{\mathcal{G}}$ be its associated PSAT instance constructed from $\mathcal{G}$ as above. Then, $\mathcal{G}$ is coherent if, and only if, $\Sigma_{\mathcal{G}}$ is satisfiable.* □

Proof Suppose $\mathcal{G}$ coherent. There exists a probability distribution P over the set of equilibria $E_G = \{e_1, \ldots, e_M\}$ such that $\sum_{j=1}^{M} P(e_j) = 1$ and that satisfies Π. Since each equilibrium is associated with a partial valuation that takes value 1 in the atoms X_i for which the associated action α_i is within the equilibrium and 0 otherwise, we consider the probability distribution over partial valuations as the probability distribution over equilibria, taking probability 0 to those partial valuations which are not associated to equilibria. This probability distribution makes $\Sigma_{\mathcal{G}}$ satisfiable.

Now, suppose $\Sigma_{\mathcal{G}}$ satisfiable. As the probability distribution that satisfies $\Sigma_{\mathcal{G}}$ makes $P(\varphi) = 1$, for all $\varphi \in \Phi_G$, it has nonzero value only on partial valuations related to equilibria. Since it also satisfies Π, considering this distribution as a probability distribution over equilibria, we find $\mathcal{G}$ coherent. ■

Note that, since PSAT is in NP, it follows from Theorem 24 that PCE-Coherence is also in NP. In other words, Theorem 22 can be seen as a corollary of Theorem 24.

Corollary 4 PCE-Coherence *over a GNP-class is polynomial time reducible to PSAT.* □

Semantics modulo satisfiability naturally underlies such context of reducing instances of PCE-Coherence to instances of PSAT. Since the partial valuations $v \in \mathbf{Val}^{\mathcal{P}}_{\Phi_G}$ stand for the equilibria of a game G, they are the only ones that should be used to evaluate the formulas $X_i \in \mathcal{P}$ standing for particular actions of players. The formulas $X_i \in \mathcal{P}$ are not, in general, consequences from Φ_G, but are also not necessarily impossibilities and may have positive probability.

Example 14 We show the reduction of PCE-Coherence to PSAT matrix format (2.5) for the observable game $\mathcal{G}_2 = \langle G_2, \Pi_2 \rangle$ with G_2 in Example 11 and Π_2 a set of PCE consisting of vector p below. Let $\varphi_{G_2} = \bigwedge \Phi_{G_2}$; we omit the columns of matrix A that are valuations which do not satisfy φ_{G_2} computed in Example 13. So, the columns in matrix A codify the five pure Nash equilibria in G_2; its last line stands for $\sum \pi_i = 1$.

$$A\pi = \begin{array}{c} a^1 \\ a^2 \\ a^3 \\ b^1 \\ b^2 \\ b^3 \\ c^1 \\ c^2 \\ c^3 \\ \varphi_{G_2} \end{array} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \\ \pi_5 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.9 \\ 0 \\ 0.5 \\ 0.5 \\ 0 \\ 0.8 \\ 0.2 \\ 0 \\ 1 \\ 1 \end{bmatrix} = p$$

This PSAT instance is satisfiable due to, for example, the vector $\pi = [0.1, 0.2, 0, 0.4, 0.3]'$, so the PCE-Coherence instance is coherent. □

5.3.1 An Algorithm for PCE-Extension

Let us turn to the PCE-Extension problem. Given a coherent observable game $\mathcal{G}$, our aim is to find the maximum and minimum observer's probabilistic constraints for some action α maintaining coherence. In other words, we need to search among the NP-witnesses of $\mathcal{G}$ for some that maximizes and minimizes the constraints on α. One might wonder whether there are polynomial time (additive) approximation algorithms for such problem, i.e., given a PCE-Extension instance consisting of $\mathcal{G}$ and α and a precision $\varepsilon > 0$, whether there exist polynomial time algorithms which return m and M such that:

- $|\underline{P}(\alpha) - m| < \varepsilon$;
- $|\overline{P}(\alpha) - M| < \varepsilon$.

The next results show the answer is negative, unless a huge breakthrough in complexity theory is achieved. First we establish an auxiliary reduction: from a game $G = \langle P, N, A, u\rangle$, we build the game $G^* = \langle P, N, A^*, u^*\rangle$, where $A_1^* = A_1 \cup \{b\}$, with $b \notin A_1$, and $A_i^* = A_i$, for $i \in P \setminus \{1\}$. Profiles $a \in A \subseteq A^*$ remain with the same utilities $u_i^*(a) = u_i(a)$, for all $i \in P$, and new profiles $= \langle b, a_2, \ldots, a_n\rangle \in A^*$, have utilities $u_1^*(p_b) = \max\{u_1(a_1', a_2, \ldots, a_n) \mid a_1' \in A_1\}$ and $u_i^*(p_b) = \max\{u_i(a) \mid a \in A\}$, for $i \in P \setminus \{1\}$.

Lemma 13 *Game G^* may be built from a game G in polynomial time and has the new pure Nash equilibria p_b in addition to the ones G already has.* □

Proof Game G^* may be built in polynomial time because for every partitioning set $\{\langle x, a_2, \ldots, a_n\rangle \mid x \in A_1\}$ of action profiles of G, we may add one unique new action profile $p_b = \langle b, a_2, \ldots, a_n\rangle$; then it is necessary to add to G^* less new utility values than the description of G already has. Let $a = \langle a_1, \ldots, a_n\rangle \in A$ be an action profile. If a is a pure Nash equilibrium of G, players in $P \setminus \{1\}$ cannot increase their utilities by choosing other action in A_i^* and, if player 1 were able to do so, it would have to be by choosing action b, then $u_1^*(b, a_2, \ldots, a_n) > u_1(a_1', a_2, \ldots, a_n)$,

for all $a'_1 \in A_1$, contradicting the definition of u^*_1. If a is not a pure Nash equilibrium of G, all players can increase their utilities by choosing other actions in A_i. Then, all pure Nash equilibria in G remains pure Nash equilibria in G^*. Finally, action profiles $p_b = \langle b, a_2, \ldots, a_n\rangle$ are clearly pure Nash equilibria in G^* and we have the result. ∎

Theorem 25 *Unless $P = NP$, there does not exist a polynomial time algorithm that approximates, to any precision $\varepsilon \in (0, \frac{1}{2})$, the expected value by the minimization version of* PCE-Extension. □

Proof Deciding the existence of pure Nash equilibria for games in GNP_k is an NP-complete problem; let us reduce this problem to PCE-Extension. Given a game G, we consider the coherent observable game $\mathcal{G} = \langle G^*, \{P(a_i) \geq 0\}\rangle$, for some arbitrary $a_i \in A_i$, for $i \in P$, together with action b as an instance of PCE-Extension. The reduction from G to $\mathcal{G}$ may be computed in polynomial time by Lemma 13. Suppose there exists a polynomial time algorithm that approximates to precision $\varepsilon \in (0, \frac{1}{2})$ the expected value by the minimization version of PCE-Extension. If G does not have any pure Nash equilibrium, all equilibria in G^* are of the type $p_b = \langle b, a_2, \ldots, a_n\rangle$, then $\underline{P}(b) = 1$ and the supposed algorithm should return $m > 1 - \varepsilon > \frac{1}{2}$. On the other hand, if G has some pure Nash equilibrium, $\underline{P}(b) = 0$ and the supposed algorithm should return $m < 0 + \varepsilon < \frac{1}{2}$. Therefore, such algorithm decides an NP-complete problem in polynomial time and $P = NP$. ∎

Theorem 26 *Unless $P = NP$, there does not exist a polynomial time algorithm that approximates, to any precision $\varepsilon \in (0, \frac{1}{2})$, the expected value by the maximization version of* PCE-Extension. □

Proof Deciding the existence of pure Nash equilibria for games in GNP_k is an NP-complete problem; let us reduce this problem to some instances of PCE-Extension. Given a game G, we consider the coherent observable game $\mathcal{G} = \langle G^*, \{P(a_i) \geq 0\}\rangle$, for some arbitrary $a_i \in A_i$, for $i \in P \setminus \{1\}$, together with all actions $a_1 \in A_1$ as $|A_1|$ different instances of PCE-Extension. The reduction from G to $\mathcal{G}$ may be computed in polynomial time by Lemma 13. Suppose there exists a polynomial time algorithm that approximates to precision $\varepsilon \in (0, \frac{1}{2})$ the expected value by the maximization version of PCE-Extension. If G does not have any pure Nash equilibrium, all equilibria in G^* are of type $p_b = \langle b, a_2, \ldots, a_n\rangle$, then $\overline{P}(a_1) = 0$, for all $a_1 \in A_1$, and the supposed algorithm should return $M < 0 + \varepsilon < \frac{1}{2}$, for all PCE-Extension instances concerning $a_1 \in A_1$. On the other hand, if G has some pure Nash equilibrium, $\overline{P}(a_1) = 1$, for some $a_1 \in A_1$, and the supposed algorithm should return $M > 1 - \varepsilon > \frac{1}{2}$, for a particular PCE-Extension instance concerning some $a_1 \in A_1$. Therefore, we are able to decide the existence of a pure Nash equilibrium in game G by running the supposed algorithm $|A_1|$ times in the instances comprehending $\mathcal{G}$ and $a_1 \in A_1$; G has a pure Nash equilibrium, if it returns $M > \frac{1}{2}$ for some instance, and G has no equilibrium otherwise. Such routine based on the supposed algorithm decides an NP-complete problem in polynomial time, hence $P = NP$. ∎

We now describe a deterministic algorithm for solving PCE-Extension whose complexity burden is all due to PCE-Coherence. Given a precision $\varepsilon = 2^{-k}$, the algorithm works by making a binary search through the binary representation of the possible constraints to α, solving PCE-Coherence in each step.

Algorithm 9 presents the procedure to solve the maximization version of PCE-Extension. We called *PCECoherence*(G, Π) the process that decides a PCE-Coherence instance $\mathcal{G} = \langle G, \Pi\rangle$.

An algorithm for solving the minimization version of PCE-Extension is easily adaptable from Algorithm 9.

Algorithm 9 PCE-Extension-BS: a PCE-Extension solver via Binary Search

Input: A coherent PCE-Coherence instance $\mathcal{G} = \langle G, \Pi \rangle$, an action $a_i \in A_i$ and a precision $\varepsilon > 0$.
Output: Maximum $P(a_i)$ value with precision ε.

```
1: k := ⌈|log ε|⌉;
2: j := 1, v_min := 0, v_max := 1;
3: if PCECoherence(G, Π ∪ {P(a_i) = 1}) = Yes then
4:         v_min := 1;
5: else
6:         while j ≤ k do
7:                 v_max = v_min + 1/2^j;
8:                 if PCECoherence(G, Π ∪ {P(a_i) ≥ v_max}) = Yes then
9:                         v_min := v_max;
10:                end if
11:                j++;
12:        end while
13: end if
14: return v_min;
```

For instance, suppose the goal is to find the maximum possible value for constraining α: the first iteration consists of solving PCE-Coherence for $P(\alpha) = 1$, if it is coherent, $\overline{P}(\alpha) = 1$, if not, $\overline{P}(\alpha) = 0$ with precision $2^0{=}1$. In case the former iteration was not coherent, the second iteration consists of solving PCE-Coherence for $P(\alpha) = 0.5$, if it is coherent, $\overline{P}(\alpha) = 0.5$, if not, $\overline{P}(\alpha) = 0$, both cases with precision $2^{-1} = 0.5$. One more iteration will give precision $2^{-2} = 0.25$ and it consists of solving PCE-Coherence for $P(\alpha) = 0.75$ in case the former iteration was coherent, or for $P(\alpha) = 0.25$ in case it was not. The process continues until the desired precision is reached and it takes $|\log 2^{-k}| + 1 = k + 1$ iterations to be completed.

Theorem 27 *Given a precision $\varepsilon > 0$,* PCE-Extension *can be obtained with $O(|\log \varepsilon|)$ iterations of* PCE-Coherence. □

Example 15 Suppose we have an observable game $\mathcal{G}_3 = \langle G_2, \Pi_3 \rangle$ with G_2 as in Example 11 and Π_3 a set of PCE consisting only of $P(a^2) = 0.9$. In order to solve PCE-Extension for finding $\overline{P}(b^2)$ with precision 2^{-6}, it will be necessary to solve seven instances of PCE-Coherence in the form below.

$$\begin{aligned} \pi_2 + \pi_4 + \pi_5 &= 0.9 \\ \pi_2 + \pi_5 &= p_5 \\ \pi_1 + \pi_2 + \pi_3 + \pi_4 + \pi_5 &= 1 \\ \pi_1, \pi_2, \pi_3, \pi_4, \pi_5 &\geq 0 \end{aligned}$$

The necessary iterations of PCE-Coherence are displayed in Table 5.3. Our algorithm returns $\overline{P}(b^2) \approx 0.890625$, which is accurate within precision $2^{-6} = 0.015625$, since $\overline{P}(b^2) = 0.9$.

Iteration	p_5	π'	Coherence
1	$1_2 = 1$	-	No
2	$0.1_2 = 0.5$	$[0.1, 0.5, 0, 0.4, 0]$	Yes
3	$0.11_2 = 0.75$	$[0.1, 0.75, 0, 0.15, 0]$	Yes
4	$0.111_2 = 0.875$	$[0.1, 0.875, 0, 0.025, 0]$	Yes
5	$0.1111_2 = 0.9375$	-	No
6	$0.11101_2 = 0.90625$	-	No
7	$0.111001_2 = 0.890625$	$[0.1, 0.890625, 0, 0.009375, 0]$	Yes

Table 5.3: *Iterations for solving* PCE-Extension *in Example 15*

5.3.2 Generalized Constraints on Equilibria

Observable game $\mathcal{G}_1$ in Examples 10 and 12 seems to imply that the formula $X_a^2 \to X_b^3$ holds, that is $\Phi_{G_1} \models_{\text{CPL}} X_a^2 \to X_b^3$, which forces $P(X_a^2 \to X_b^3) = 1$ and thus $P(X_a^2) \leq P(X_b^3)$. Thus, the propositional theory derived from a game that will reach an equilibrium may offer a deeper understanding of such game for an external observer and motivates the following generalization of our goal problems.

Given a game G, consider the set of propositional variables $\mathcal{P}$ in one-to-one correspondence with all actions $\cup_{i \in P} A_i$, where each variable $X_i^j \in \mathcal{P}$ represents the occurrence of action $a_i^j \in A_i$ in the equilibrium. A formula φ, with $\text{Var}(\varphi) \subseteq \mathcal{P}$, describes a combination of such statements at equilibrium. On the semantic side, we identify each pure equilibrium e with a partial valuation v_e over $\mathcal{P}$ such that for every action $a_i^j \in A_i$, $v_e(X_i^j) = 1$ iff $a_i \in e$. So, a formula φ is satisfied at equilibrium e, represented as $\varphi \in e$, if $v_e(\varphi) = 1$.

We can generalize the notion of PCE in (5.1) as a set

$$\Pi = \Big\{ P(\varphi_k) \ \bowtie_k \ p_k \ \Big| \ \bowtie_k \in \{\leq, \geq, =\}, \ 1 \leq k \leq K \Big\},$$

where φ_k are formulas such that $\text{Var}(\varphi_k) \subseteq \mathcal{P}$, so instead of restricting the probabilities of actions at equilibrium, we can now describe the probabilities of compound logical statements at equilibrium.

Definition 9 (Generalized PCE Coherence and Extension Problems) Given an observable game $\mathcal{G} = \langle G, \Pi \rangle$ with a set Π of generalized PCE, *Generalized PCE Coherence Problem* consists of deciding if it is *coherent*, that is if there exists a probability function over the set of G-equilibria that satisfies all constraints in Π. And the *Generalized PCE Extension Problem* for a coherent observable game with a generalized set PCE Π and a formula ψ, with $\text{Var}(\psi) \subseteq \mathcal{P}_G$, consists of finding upper and lower bounds for $P(\psi)$ that satisfy Π. □

Note that a generalized PCE instance $\Pi = \{P(\varphi_i) \bowtie_i p_i, 1 \leq i \leq K\}$ may be reduced to a PSAT instance analogously to the reduction in Section 5.3.

5.4 Coherence Allowing Mixed Equilibria

We proceed on studying PCE-Coherence with respect to the more general concept of mixed Nash equilibrium. A *strategy* for player i is a rational probability distribution σ_i over the set A_i of actions for player i and $\Sigma = \Sigma_1 \times \cdots \times \Sigma_n$ is the set of *strategy profiles*, in which each Σ_i is the set of all possible strategies for player i. The set of actions with nonzero probability in a strategy σ_i is its

support. We call *pure strategy* a strategy with unitary support and from now on we identify actions with pure strategies; in contrast we call *mixed strategy* a strategy that is not a pure strategy.

It is assumed that each player's choice of strategy is independent from all other players' choices, so the *expected utility function* U_i for player i is given by:

$$U_i(\sigma) = \sum_{a \in A} u_i(a) \prod_{j \in P} \sigma_j(a_j),$$

where $\sigma \in \Sigma$, and $a = \langle a_1, \ldots, a_n \rangle$ with $a_j \in A_j$. A strategy profile $e = \langle \sigma_1, \ldots, \sigma_i, \ldots, \sigma_n \rangle$ is a *mixed (Nash) equilibrium* if, for every player i, $U_i(e) \geq U_i(\sigma_1, \ldots, \sigma_i', \ldots, \sigma_n)$, for every $\sigma_i' \in \Sigma_i$; each σ_i in e is called a *best response* for player i given the other players strategies in e. Then, a strategy profile is a mixed Nash equilibrium if, and only if, it is composed by best responses for all players. A game G always has at least one mixed Nash equilibrium (Nash, 1951).

Note that an action profile a may be seen as a strategy profile σ by taking each action $a_i \in a$ for player i as the strategy $\sigma_i \in \sigma$ that assigns 1 to a_i and 0 to the other actions in A_i. This way, a is a pure Nash equilibrium if, and only if, its associated σ is a mixed Nash equilibrium.

Mixed strategies may be better understood if we think of a game situation that repeatedly occurs and, in each instance, the players choose their actions randomly according to their mixed strategies. In this context, an observable game is a game that repeatedly occurs and which is known to be at one of its (mixed) equilibria, but the external observer does not know exactly which one. An instance of the game will be played and the observer assigns subjective probabilities to actions being part of the action profile to be reached in that instance. Formally, an *observable game* is a pair $\mathcal{G} = \langle G, \Pi \rangle$ as before. We may again interpret these probability assignments as bets placed by the observer to the actions that the players are allowed to choose.

Another way of understanding observable games is by imagining there are many game situations with the same setting and that repeatedly occurs and all these game situations are at some mixed Nash equilibrium. With that knowledge, the external observer looks at one game situation that is about to have a new instance played, but he does not know exactly which game situation among the many ones existing this is. Then, the observer does not know which is the mixed Nash equilibrium the game situation he is looking at is in and he assigns subjective probabilities for the players' actions being part of the action profile resulting from the game situation instance.

As before, we suppose that there is a probability distribution over the mixed Nash equilibria. It is important to note that this probability distribution is independent from probability distribution in a mixed strategy. The former probability distribution ranges over mixed Nash equilibria and the latter ranges over actions. If $e_j = \langle \sigma_{1j}, \ldots, \sigma_{nj} \rangle \in E_G$, σ_{ij} designates the i-th component of e_j. In this setting, the probability function P over mixed equilibria induces the probability $P(a_i)$ of an action a_i by

$$P(a_i) = \sum_{j \,|\, e_j \in E_G} \sigma_{ij}(a_i) \cdot P(e_j).$$

The definition of *probabilistic constraints on equilibria (PCE)* is analogous to that in Section 5.2, namely a set of probability assignments on actions. PCE-Coherence is similarly defined as the problem of, given an observable game $\mathcal{G} = \langle G, \Pi \rangle$, deciding if it is *coherent*, i.e. deciding if there exists a probability function over the set of equilibria that satisfies all constraints in Π. PCE-Extension is also completely analogous to the one in the pure equilibrium setting.

Example 16 Recall Example 10 where we had the game between Alice and Bob in which Alice's actions were a^1, a^2 and a^3 and Bob's actions were b^1, b^2 and b^3, such that the joint utilities are in Table 5.1. This game has three pure equilibria, now viewed as special cases of mixed equilibria: $e_1 = \langle \sigma_1^1, \sigma_2^1 \rangle$, where $\sigma_1^1(a^1) = 1$, $\sigma_1^1(a^2) = \sigma_1^1(a^3) = 0$, $\sigma_2^1(b^1) = 1$, $\sigma_2^1(b^2) = \sigma_2^1(b^3) = 0$; and $e_2 = \langle \sigma_1^2, \sigma_2^2 \rangle$ and $e_3 = \langle \sigma_1^3, \sigma_2^3 \rangle$ that are also based on the ones described in Example 10. However, there are several other mixed equilibria among which we highlight $e_4 = \langle \sigma_1^4, \sigma_2^4 \rangle$ given by

$$\sigma_1^4(a^1) = \frac{2}{3}, \quad \sigma_1^4(a^2) = \frac{1}{3}, \quad \sigma_1^4(a^3) = 0, \qquad \sigma_2^4(b^1) = \frac{4}{5}, \quad \sigma_2^4(b^2) = \frac{1}{5}, \quad \sigma_2^4(b^3) = 0.$$

We have established that if only pure equilibria are considered, the constraints $P(a^2) = \frac{1}{3}$ and $P(b^3) = \frac{1}{4}$ are incoherent. However, in a context that considers mixed Nash equilibria, they are coherent, as we detail in Example 17. □

5.4.1 Classes of Games Allowing Mixed Equilibria

It is not known if there exists a nondeterministic polynomial algorithm that computes an exact mixed Nash equilibrium for games with at least three players and such a result would imply theoretical breakthroughs in complexity theory (Etessami and Yannakakis, 2010). On the other hand, for games with two players, such an algorithm may be described; the following result helps to elucidate the problem and its combinatorial nature.

Proposition 5 (Papadimitriou (2007)) *A mixed strategy is a best response if and only if all pure strategies in its support are best responses.* □

Thus, in order to compute a mixed Nash equilibrium $\sigma \in \Sigma$, it is first necessary to establish all the supports for all the players. We write $U_i(\sigma|a_i)$ for player i's expected utility for the strategy profile $\sigma' = \langle \sigma_1, \ldots, a_i, \ldots, \sigma_n \rangle$ (remember we identify actions and pure strategies). Given the supports with size k_i for each player i, there are $k_i - 1$ equations on the other players' strategies in σ stating that the k_i player i's expected utilities $U_i(\sigma|a_i)$, for his pure strategies a_i in the support of σ_i are equal. Then, for σ to be a mixed Nash equilibrium:

- The probabilities in σ must satisfy the system of $\sum_{i \in P}(k_i - 1)$ equations described above;
- For each player i, his expected utilities $U_i(\sigma|a_i)$, for the pure strategies a_i in the support of σ_i, must have value at least as the expected utilities $U_i(\sigma|a_i)$, for the pure strategies a_i not in the support.

In case σ_i is a pure strategy, there are no equations associated to player i in the system and in case the game has only two players, it becomes a linear system. A nondeterministic polynomial algorithm for computing mixed Nash equilibria for 2-player games consists of guessing supports, then verifying if the corresponding linear system has a solution and, if so, verifying if the derived strategies obey Proposition 5.

Thus, in order to generate a GNP-class with respect to mixed Nash equilibrium, let 2GNP be the class of games with two players; further, we have to restrict the equilibrium concept to "small" representations, as there are infinitely many mixed Nash equilibria for some games — i.e. the equilibria representation sizes will be polynomially bounded on the game parameters. Note that the restriction on "small" representations implies that each game has only finitely many mixed equilibria.

We call 2G the class of games similarly defined but without the restriction on the representations of equilibria; then, 2G fails to be a GNP-class. Both standard normal form and graphical normal form may be used to represent games in 2GNP and 2G. In next section, we present a logic-based algorithm for computing equilibria for 2-player games.

5.4.2 Computing Mixed Nash Equilibria via Ł$_\infty$-SAT

We now build a set of formulas Φ_G in polynomial time from a 2-player game G such that the valuations that satisfy Φ_G encode the Nash equilibria of G.

Let $P = \{a, b\}$, $A = A_a \times A_b$, with $A_a = \{a^1, \ldots, a^{n_a}\}$ and $A_b = \{b^1, \ldots, b^{n_b}\}$, and $u = \langle u_a, u_b\rangle$ with $u_a : A_a \to \mathbb{Q}$ and $u_b : A_b \to \mathbb{Q}$. Let $\sigma = \langle \sigma_a, \sigma_b\rangle$ be a generic strategy profile with $\sigma_a : A_a \to [0,1] \cap \mathbb{Q}$ and $\sigma_b : A_b \to [0,1] \cap \mathbb{Q}$ generic mixed strategies for the players a and b. To each probability $\sigma_a(a^i)$ and $\sigma_b(b^j)$ we associate propositional variables X_a^i and X_b^j respectively. Thus, $X_a^1, \ldots, X_a^{n_a}, X_b^1, \ldots, X_b^{n_b} \in \Phi_G$. Let us build the formulas of Φ_G according to player a; the formulas according to player b are analogous.

To assure the propositional variables represent probabilities, we add to Φ_G the formulas:

(i) $X_a^1 \oplus \cdots \oplus X_a^{n_a}$;

(ii) $\neg(X_a^1 \odot X_a^2)$, $\neg[(X_a^1 \oplus X_a^2) \odot X_a^3]$, $\ldots$, $\neg[(X_a^1 \oplus \cdots \oplus X_a^{n_a-1}) \odot X_a^{n_a}]$.

These formulas are built in time $O(n_a^2)$.

Let us denote by $\ulcorner U_a(\sigma|a^i) = U_a(\sigma|a^k)\urcorner$ the formula that only has truth value 1 for a valuation v that encodes strategy profile σ for which player a's expected utilities $U_a(\sigma|a^i)$ and $U_a(\sigma|a^k)$ are equal. For each player a's action a^i, we build the formula:

$$\left(\bigoplus_{k=1,\ldots,n} \ulcorner U_a(\sigma|a^i) = U_a(\sigma|a^k)\urcorner \odot X_a^k\right) \odot X_a^i,$$

which we denote by $\zeta_a(a^i)$. Let v be a valuation satisfying formulas (i) and (ii) and, therefore, representing strategy profile σ. Then, the formula $\zeta_a(a^i)$ is evaluated by v with the exact same value as X_a^i if one of three following cases occurs: a^i is not in the strategy σ_a support; a^i is the only action in σ_a support (σ_a is a pure strategy); the expected utility $U_a(\sigma|a^i)$, for player a's pure strategy a^i, which is in σ_a support, is equal to the expected utilities $U_a(\sigma|a^k)$, for all a^k in σ_a support. If neither of these cases occur, v evaluates $\zeta_a(a^i)$ strictly less than it evaluates X_a^i. We add to Φ_G the following formula, that has truth value 1 only if all the pure strategies a^i fall into one of the three cases just described:

(iii) $\zeta_a(a^1) \oplus \cdots \oplus \zeta_a(a^{n_a})$.

We denote by $\ulcorner U_a(\sigma|a^i) \leq U_a(\sigma|a^k)\urcorner$ the formula that only has truth value 1 for a valuation v that encodes the strategy profile σ for which player a's expected utility $U_a(\sigma|a^i)$ is at most his expected utility $U_a(\sigma|a^k)$. For each player a's pure strategy a^i we build formula

$$\bigoplus_{k=1,\ldots,n} \ulcorner U_a(\sigma|a^i) \leq U_a(\sigma|a^k)\urcorner \odot X_a^k,$$

which we denote by $\chi_a(a^i)$. Let v be a valuation satisfying all the formulas (i), (ii) and (iii) in Φ_G. The formula $\chi_a(a^i)$ only has value 1 if player a's expected utility $U_a(\sigma|a^i)$, for pure strategy a^i, is at most the expected utilities $U_a(\sigma|a^k)$, for all her pure strategies a^k in σ_a support. We also add to Φ_G all the following formulas:

(iv) $\chi_a(a^1), \ldots, \chi_a(a^{n_a})$.

Assuming formulas $\ulcorner U_a(\sigma|a^i) = U_a(\sigma|a^k)\urcorner$ and $\ulcorner U_a(\sigma|a^i) \leq U_a(\sigma|a^k)\urcorner$ are built in time $O(p(|G|))$ on size $|G|$ of an instance G of a game, formula (iii) and formulas (iv) are built in time $O(n_a^2 \cdot p(|G|))$.

By the discussion in Section 5.4.1, if a valuation v encoding $\sigma = \langle \sigma_a, \sigma_b \rangle$ satisfies the set Φ_G as we have built, then σ_a is a best response and, as we analogously add to Φ_G all the formulas (i)-(iv) concerning player b, σ_b is also a best response. In that case, σ is a Nash equilibrium and we state the following result.

Theorem 28 *Given a 2-player game and a strategy profile σ encoded by Ł$_\infty$-valuation v as in previous discussion, σ is a Nash equilibrium if, and only if, v satisfies Φ_G.* □

We still need to explicitly write the formulas $\ulcorner U_a(\sigma|a^i) = U_a(\sigma|a^k)\urcorner$ and $\ulcorner U_a(\sigma|a^i) \leq U_a(\sigma|a^k)\urcorner$. Player a's expected utilities $U_a(\sigma|a^i)$, for pure strategies a^i, are given by:

$$U_a(\sigma|a^i) = \sum_{j=1}^{n_b} u_a(a^i, b^j)\sigma_b(b^j).$$

Thus, our goal is to represent linear equations and inequalities in Ł$_\infty$ with variables $\sigma_b(b^j)$ and rational coefficients $u_a(a^i, b^j)$. By representing an equation (or an inequality) in Ł$_\infty$ we mean building a formula or a set of formulas that is satisfied by a valuation v if, and only if, v encodes a solution to the equation (or inequality). Let us treat the general case of an equation

$$\gamma_1 x_1 + \cdots + \gamma_n x_n = 0, \tag{5.3}$$

with variables x_i and rational fractions γ_i, for which we are interested in solutions in $[0,1]^n$.

Before building the representation, we put this equation in an equivalent format that we define in two steps. First we turn (5.3) into

$$\sum_{i \in I} \frac{\tilde{\gamma}_i}{m} x_i = \sum_{j \in J} \frac{\tilde{\gamma}_j}{m} x_j,$$

where $i \in I$, if $\gamma_i \geq 0$, and $j \in J$, if $\gamma_j < 0$, with $I \cup J = \{1, \ldots, n\}$; m is the least common multiple of all denominators in fractions γ_k, for $k = 1, \ldots, n$; $\frac{\tilde{\gamma}_i}{m}$ are equivalent fractions to γ_i, for $i \in I$; and $\frac{\tilde{\gamma}_j}{m}$ are equivalent fractions to $-\gamma_j$, for $j \in J$. Note that $\tilde{\gamma}_k$ and m are positive integers, for $k = 1, \ldots, n$. The second step consists in turning (5.3) into

$$\sum_{i \in I} \frac{\tilde{\gamma}_i}{\mu} x_i = \sum_{j \in J} \frac{\tilde{\gamma}_j}{\mu} x_j,$$

where $\mu = \max\{\tilde{\gamma}_1, \ldots, \tilde{\gamma}_n\}$. In the final format, both sides of the equation take values in $[0,1]$ for any vector $\langle x_1, \ldots, x_n \rangle \in [0,1]^n$.

To each equation variable x_k, we associate a propositional variable X_k, for $k = 1, \ldots, n$, and build the following set of formulas that represent (5.3), using the auxiliary propositional variables $C_{\frac{1}{M}}$ and $\tilde{X}_k$, for $k = 1, \ldots, n$:

$$\bigoplus_{i \in I} \tilde{\gamma}_i \tilde{X}_i \leftrightarrow \bigoplus_{j \in J} \tilde{\gamma}_j \tilde{X}_j; \tag{5.4}$$

$$C_{\frac{1}{\mu}} \leftrightarrow \neg(\mu - 1) C_{\frac{1}{\mu}}; \tag{5.5}$$

$$\tilde{X}_k \rightarrow C_{\frac{1}{\mu}}; \tag{5.6}$$

$$X_k \leftrightarrow \mu \tilde{X}_k, \text{ for } k = 1, \ldots, n. \tag{5.7}$$

Let valuation v evaluate formulas (5.5)-(5.7) with truth value 1. Then, it also evaluates formula (5.4) with truth value 1 if, and only if, $\langle v(X_1), \ldots, v(X_n) \rangle$ satisfies equation (5.3). Formula (5.5) makes variable $C_{\frac{1}{\mu}}$ have value $\frac{1}{\mu}$ (for any v); formula (5.6) guarantees that variable $\tilde{X}_k$ has value at most $\frac{1}{\mu}$ and, together with formula (5.7), makes it have truth value exactly $\frac{v(X_k)}{\mu}$.

We are now able to explicitly write formulas $\ulcorner U_a(\sigma|a^i) = U_a(\sigma|a^k) \urcorner$, remembering that we identified equation variables $\sigma_b(b^j)$ with propositional variables X_b^j. We may take $\ulcorner U_a(\sigma|a^i) = U_a(\sigma|a^k) \urcorner$ as the maximum — the Ł$_\infty$-operator $\wedge$ — of formulas (5.4)-(5.7), or even as only formula (5.4) and add formulas (5.5)-(5.7) to the set Φ_G. To explicitly write formulas $\ulcorner U_a(\sigma|a^i) \leq U_a(\sigma|a^k) \urcorner$ we use *mutatis mutandis* this very same technique considering

$$\gamma_1 x_1 + \cdots + \gamma_n x_n \leq 0$$

instead of equation (5.3) and using

$$\bigoplus_{i \in I} \tilde{\gamma}_i \tilde{X}_i \rightarrow \bigoplus_{j \in J} \tilde{\gamma}_j \tilde{X}_j$$

instead of formula (5.4).

Let us discuss the time complexity of building the set Φ_G. By the observations we have been doing throughout this section, computing set Φ_G may be done in polynomial time if computing formulas $\ulcorner U_a(\sigma|a^i) = U_a(\sigma|a^k) \urcorner$ and $\ulcorner U_a(\sigma|a^i) \leq U_a(\sigma|a^k) \urcorner$ also may be done in polynomial time. The equations and inequalities in terms of expected utilities U_a and U_b that we have discussed so far may surely be derived from a 2-player game in polynomial time. However, writing formulas (5.4), (5.5) and (5.7) takes exponential time in the binary representation of $\tilde{\gamma}_k$, $\mu - 1$ and μ due to n-fold Ł$_\infty$-conjunctions in expressions $n\psi$, for $n \in \{\tilde{\gamma}_k, \mu - 1, \mu\}$. This situation may be circumvented by taking advantage of binary representation in the same way was done in Section 3.4.2. We need to replace every expression $n\psi$, where $n \in \mathbb{N} \setminus \{0, 1\}$, by the corresponding one as in (3.11) and add the corresponding formulas in (3.12) to the original collection (5.4)-(5.7). Therefore, we may represent in Ł$_\infty$ linear equations or inequalities (with rational coefficients given in binary representation) in polynomial time. Then, Φ_G may be built from G in polynomial time.

A nondeterministic polynomial algorithm for computing mixed Nash equilibria for games in 2GNP or in 2G consists of the reduction from the input game $G \in$ 2GNP to its corresponding set Φ_G, with $\mathrm{Var}(\Phi_G) \subseteq \mathcal{P}$, where $\mathcal{P}$ is a finite set, followed by a nondeterministic polynomial algorithm computing partial valuations $v \in \mathbf{Val}^{\mathcal{P}}$ encoding the equilibria, that is $v \in \mathbf{Val}^{\mathcal{P}}_{\Phi_G}$. In case only

equilibria with bounded representation size are considered, one should bound the representation size of truth values computed by the $Ł_\infty$-solver employed.

5.5 From PCE-Coherence to ŁIPSAT

Let us also explore a linear algebraic formulation of PCE-Coherence in the mixed equilibrium setting. Let $\Pi = \{P(\alpha_i) \bowtie_i p_i, 1 \leq i \leq K\}$ be a set of PCE for an observable game with M mixed Nash equilibria. PCE-Coherence becomes the problem of deciding the existence of a vector π satisfying

$$\begin{array}{rcl} A\pi & \bowtie & p \\ \sum \pi_j & = & 1 \\ \pi & \geq & 0 \end{array} \tag{5.8}$$

where $A = [a_{ij}]$ is a $K \times M$ matrix whose columns represent the mixed Nash equilibria in the game. In this case $a_{ij} = \sigma_{pj}(\alpha_i)$, where $p \in P$ is such that $\alpha_i \in A_p$, that is a_{ij} is the probability assignment of action α_i in the mixed equilibrium e_j by player p. The existence of small witnesses for coherence given by Carathéodory's Theorem (Proposition 1) also applies in this setting. This leads to a similar complexity result for PCE-Coherence as we have in Theorem 22.

Theorem 29 PCE-Coherence *over a GNP-class is a problem in NP.* □

Proof This proof is totally analogous to that of Theorem 22. Suppose the observable game $\mathcal{G} = \langle G, \Pi \rangle$ is coherent and $|\Pi| = K$. Therefore, there exists a probability distribution π over the set of all possible (mixed) Nash equilibria that satisfy Π. By the Carathéodory's Theorem (Proposition 1), there is a probability distribution assigning nonzero probabilities to at most $K+1$ equilibria. These equilibria are polynomially bounded in size since G is in a GNP-class. Therefore, there is a witness π whose size is polynomially bounded attesting Π is satisfied, so PCE-Coherence is in NP. ■

Corollary 5 PCE-Coherence *over* 2GNP *is a problem in NP.* □

Example 17 We show the reduction of PCE-Coherence to the matrix form (5.8) for observable game in Example 16. We only show the columns of matrix A corresponding to equilibria mentioned in Example 16, which already provides a probability distribution satisfying the set of PCE in vector p below; last line in A stands for $\sum \pi_i = 1$.

$$A\pi = \begin{array}{c} a^1 \\ a^2 \\ a^3 \\ b^1 \\ b^2 \\ b^3 \end{array} \begin{bmatrix} 1 & 0 & 0 & \frac{2}{3} \\ 0 & 1 & 0 & \frac{1}{3} \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & \frac{4}{5} \\ 0 & 0 & 0 & \frac{1}{5} \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \end{bmatrix} = \begin{bmatrix} p_2 \\ \frac{1}{3} \\ p_4 \\ p_5 \\ p_6 \\ \frac{1}{4} \\ 1 \end{bmatrix} = p$$

This matrix system is solvable due to, for example, the vector $\pi = [\frac{1}{2}, \frac{1}{4}, 0, \frac{1}{4}]'$, so the PCE-Coherence instance is coherent. □

An algorithm for solving PCE-Coherence has to provide a means to find a solution for (5.8) if one exists and, otherwise, determine that no solution is possible. Note that in the case of pure equilibria, the matrix A entries a_{ij} could have values only 0 and 1, but now $a_{ij} = \sigma_{pj}(\alpha_i) \in [0, 1]$. Here, such algebraic form resembles the one of ŁIPSAT in Section 4.2 and, now, we provide a reduction from PCE-Coherence to ŁIPSAT.

Let $\mathcal{G} = \langle G, \Pi \rangle$ be an observable game such that G is a 2-player game and $\Pi = \{P(\alpha_i) \bowtie_i p_i \mid 1 \leq i \leq K\}$ is a set of PCE. Let $\mathcal{P}$ be a set of propositional variables in one-to-one correspondence to all possible actions in G, denote by $X_i \in \mathcal{P}$ the propositional variable associated to each action α_i appearing in Π and let $\Pi_{\mathcal{P}} = \{P(X_i) \bowtie_i p_i \mid 1 \leq i \leq K\}$. By the polynomial reduction in Section 5.4.2, there is a set of formulas Φ_G, with $\mathcal{P} \subseteq \mathrm{Var}(\Phi_G)$, such that if we have a partial valuation $v \in \mathbf{Val}_{\Phi_G}^{\mathrm{Var}(\Phi_G)}$, then $\mathcal{P}$ is a set of propositional variables whose truth values by v represent probabilities for mixed strategies which jointly played are an equilibrium. Construct the ŁIPSAT instance $\Sigma_{\mathcal{G}} = \Pi_{\mathcal{P}} \cup \{P(\varphi) = 1 \mid \varphi \in \Phi_G\}$. If all symbols $\bowtie_i$ in $\Pi_{\mathcal{P}}$ are the equality symbol, such ŁIPSAT instance may be put in the equivalent atomic normal form $\langle \Phi_G, \Pi_{\mathcal{P}} \rangle$.

Theorem 30 *Let $\mathcal{G} = \langle G, \Pi \rangle$ be an observable game, where G is a 2-player game and Π is a set of PCE, and let $\Sigma_{\mathcal{G}}$ be its associated ŁIPSAT instance constructed from $\mathcal{G}$ as above. Then, $\mathcal{G}$ is coherent if, and only if, $\Sigma_{\mathcal{G}}$ is satisfiable.* □

Proof This proof is totally analogous to that of Theorem 24. Suppose $\mathcal{G}$ coherent. There exists a probability distribution P over the set of equilibria $E_G = \{e_1, \ldots, e_M\}$ such that $\sum_{j=1}^{M} P(e_j) = 1$ and that satisfies Π. Since each equilibrium is associated with a partial valuation, we consider the probability distribution over partial valuations as the probability distribution over equilibria, taking probability 0 to those partial valuations which are not associated to equilibria. This probability distribution makes $\Sigma_{\mathcal{G}}$ satisfiable.

Now, suppose $\Sigma_{\mathcal{G}}$ satisfiable. As the probability distribution that satisfies $\Sigma_{\mathcal{G}}$ makes $P(\varphi) = 1$, for all $\varphi \in \Phi_G$, it has nonzero value only on partial valuations related to equilibria. Since it also satisfies Π, considering this distribution as a probability distribution over equilibria, we find $\mathcal{G}$ coherent. ■

As a consequence of the above theorem, PCE-Coherence over 2GNP and 2G are polynomial time reducible to ŁIPSAT provided that, in the case of 2GNP, only valuations whose values take an adequate bounded representation size are allowed for establishing satisfiability of ŁIPSAT instances.

Corollary 6 PCE-Coherence *over* 2G *is a problem in NP.* □

The following proposition has a reduction that we use for establishing the NP-completeness of PCE-Coherence over 2GNP and 2G and the inapproximability results in next section.

Proposition 6 (Conitzer and Sandholm (2008)) *Let φ be a CNF CPL-formula with n propositional variables. Then there exists a 2-player game G_φ which may be built in polynomial time, with $f_i \in A_i$, for $i \in \{1, 2\}$, such that φ is satisfiable if, and only if, it has a mixed Nash equilibrium which is a strategy profile $\sigma_{SAT} = \langle \sigma_1, \sigma_2 \rangle$, where σ_1 assigns positive probability $\frac{1}{n}$ to n distinct actions in $A_1 \setminus \{f_1\}$. Furthermore, action profile $\sigma_f = \langle f_1, f_2 \rangle$ is the only other possible mixed Nash equilibrium in G_φ.* □

Theorem 31 PCE-Coherence *over* 2GNP *and* 2G *are NP-complete.* □

PROOF Memberships in NP are stated in Corollaries 5 and 6. CPL-SAT is an NP-complete problem; let us reduce this problem to PCE-COHERENCE. Given a CNF CPL-formula φ, let G_φ be the game in Proposition 6. We consider the instance $\mathcal{G} = \langle G_\varphi, \{P(f_1) = 0\}\rangle$ of PCE-COHERENCE, which may be computed from φ in polynomial time. $\mathcal{G}$ is coherent if, and only if, G_φ has another mixed Nash equilibrium beyond σ_f, which happens if, and only if, φ is satisfiable. Thus, PCE-COHERENCE over 2GNP and 2G are NP-hard. ■

There are many results stating that deciding on the existence of mixed Nash equilibria in 2-player games with some property, such as uniqueness, Pareto-optimality, etc, are NP-complete problems (Conitzer and Sandholm, 2008; Gilboa and Zemel, 1989). PCE-COHERENCE may be seen as an addition to this list by Theorem 31 if one glimpses it as the problem of deciding whether there are at most $K+1$ equilibria for which there is an associated vector π that satisfies conditions in (5.8).

5.5.1 PCE-EXTENSION Allowing Mixed Equilibria

We now analyze PCE-EXTENSION in analogy of what has been done in Section 5.3.1. The definition of PCE-EXTENSION is analogous to that of the pure equilibrium case, i.e. given a coherent observable game $\mathcal{G} = \langle G, \Pi\rangle$, where G is a 2-player game, and an action $a_i \in A_i$, PCE-EXTENSION consists in finding probability functions $\underline{P}$ and $\overline{P}$ that satisfy Π such that $\underline{P}(a_i)$ is minimal and $\overline{P}(a_i)$ is maximal. As far as approximation algorithms are concerned for PCE-EXTENSION, we have analogous results to Theorems 25 and 26.

Theorem 32 *Unless $P = NP$, there does not exist a polynomial time algorithm that approximates, to any precision $\varepsilon \in (0, \frac{1}{2})$, the expected value by the minimization version of* PCE-EXTENSION. □

PROOF CPL-SAT is an NP-complete problem; let us reduce this problem to PCE-EXTENSION. Given a CNF CPL-formula φ, let G_φ be the game in Proposition 6. We consider the coherent observable game $\mathcal{G} = \langle G_\varphi, \{P(a_i) \geq 0\}\rangle$, for some arbitrary $a_i \in A_i$, for $i \in P$, together with action f_1 as an instance of PCE-EXTENSION. The reduction from φ to $\mathcal{G}$ may be computed in polynomial time. Suppose there exists a polynomial time algorithm that approximates to precision $\varepsilon \in (0, \frac{1}{2})$ the expected value by the minimization version of PCE-EXTENSION. If φ is not satisfiable, the only equilibrium in G_φ is σ_f, then $\underline{P}(f_1) = 1$ and the supposed algorithm should return $m > 1 - \varepsilon > \frac{1}{2}$. On the other hand, if φ is satisfiable, $\underline{P}(f_1) = 0$ and the supposed algorithm should return $m < 0 + \varepsilon < \frac{1}{2}$. Therefore, such algorithm decides an NP-complete problem in polynomial time and $P = NP$. ■

Theorem 33 *Unless $P = NP$, there does not exist a polynomial time algorithm that approximates, to any precision $\varepsilon \in (0, \frac{1}{6})$, the expected value by the maximization version of* PCE-EXTENSION. □

PROOF CPL-SAT is an NP-complete problem; let us reduce this problem to PCE-EXTENSION. Given a CNF CPL-formula φ, let G_φ be the game in Proposition 6. We consider the coherent observable game $\mathcal{G} = \langle G_\varphi, \{P(a_2) \geq 0\}\rangle$, for some arbitrary $a_2 \in A_2$, together with some arbitrary action $a_1 \in A_1 \setminus \{f_1\}$ as an instance of PCE-EXTENSION. The reduction from φ to $\mathcal{G}$ may be computed in polynomial time. Suppose there exists a polynomial time algorithm that approximates to precision $\varepsilon \in (0, \frac{1}{6})$ the expected value by the maximization version of PCE-EXTENSION. If φ is not satisfiable, the only equilibrium in G_φ is σ_f, then $\overline{P}(a_1) = 0$ and the supposed algorithm

should return $M < 0 + \varepsilon < \frac{1}{6}$. On the other hand, if φ is satisfiable, $\overline{P}(a_1) = \frac{1}{3}$ and the supposed algorithm should return $M > \frac{1}{3} - \varepsilon > \frac{1}{6}$. Therefore, such algorithm decides an NP-complete problem in polynomial time, hence $P = NP$. ■

PCE-Extension in the mixed equilibrium setting may also be solved by Algorithm 9 with $PCECoherence(G, \Pi)$ now being a process that allows mixed equilibria. We have a similar result as in Theorem 27.

Theorem 34 *Given an instance of* PCE-Extension *over* 2GNP *or* 2G *and a precision* $\varepsilon > 0$, PCE-Extension *can be obtained with* $O(|\log \varepsilon|)$ *iterations of* PCE-Coherence. □

5.6 Some Thoughts on Game-Theoretic Modeling

We can divide the study of equilibrium problems in two fundamental aspects. On the one hand, the model *per se*; in this case, equilibrium concepts are often understood as models that explain (rational or not) agent behavior, e.g. in the markets or in a biological system. On the other hand, algorithms; such issues become relevant in the cases where it is important to actually compute an equilibrium. Coherence of observable games — or, coherence of probabilistic constraints on equilibria — does not constitute a concept of equilibrium, however we can make an analogy between their study and the two aforementioned aspects. Indeed, in this chapter, we formalized the concept of coherent observable games and solved the Coherence and Extension Problems.

The coherence of an observable game models the interaction that the uncertainty about the game should have with the knowledge that such game reaches equilibrium. Thus, an incoherent observable game explains the inevitable failure of the observer in taking advantage of his position of observer, e.g. the sure loss of the better in de Finetti's probability interpretation or the poor management of the local beer producer observing the oligopolistic market. However, it is important to notice that the observer's subjective probability assignments may be coherent and still far from reality. In this way, an incoherent observable game alone may be enough to explain the failure of the observer, but a coherent observable game is not enough to guarantee his success. All in all, the sharpness of the observer's probability assignments also depends on how deep is his knowledge about the game and to be coherent is only part of his enterprise in making a good analysis of the game he observes.

As far as equilibrium concepts are concerned, the usefulness of the actual computation of equilibria is part of an ongoing debate (Papadimitriou and Roughgarden, 2008). Since by the aspect of the model *per se*, an equilibrium concept explains agents behavior, the actual computation of an equilibrium might be regarded as completely irrelevant. Nevertheless, it might also be argued that it is only reasonable to accept that agents behave according to an equilibrium if it is not too hard to compute such equilibrium. In light of this discussion, we may conclude that the hardness results concerning PCE-Coherence point to the difficulty of the observer in being coherent; thus, it may explain failures in the management by local producers when competing with oligopolists.

However, PSAT and ŁIPSAT have been shown to have easy-hard-easy phase transition, which means that possibly most cases of PCE-Coherence over GNP-classes can be solved easily. By this hypotheses, in most cases it is not difficult for an observer to be coherent and then the responsibility for a poor management falls entirely over the poor knowledge on the oligopolistic market by the

local producer. Moreover, we believe that the framework of observable games we set in this chapter is in the interest of an observer who actually wants to compute the coherence and the extension of his probabilities over actions which are in equilibrium independently of whether this equilibrium was actually computed or how it was established; e.g. again, the local producer observing an oligopolistic market. In this way, the reductions from PCE-Coherence to PSAT and ŁIPSAT are encouraging due to their phase transition behavior and the improvement in the technologies for implementing linear algebraic solvers and CPL-SAT, PSAT and even ŁIPSAT solvers.

The following paper, which is related to this chapter, has been submitted to a journal and is currently under review.

- Sandro Preto, Eduardo Fermé and Marcelo Finger. Coherence of probabilistic constraints on Nash equilibria.

Chapter 6

Conclusions

In general, we applied semantics modulo satisfiability — a restricted semantics which comprehends only valuations that satisfy some specific set of formulas — in computationally tackling some problems in an efficient way. Evaluation via such restricted semantics was already present, although not evidenced, in the work of Finger and De Bona (2015) for solving the Probabilistic Satisfiability Problem; we defined an adequate framework for semantics modulo satisfiability, which was used throughout all this thesis. We believe we have shown how natural such concept is, first, by examining it in analogy to non-valid formulas and in parallel with propositional theories and, second, by applying it in the ways we did.

We investigated implicit representations of functions by logical formulas in the Łukasiewicz Infinitely-valued Logic by means of semantics modulo satisfiability; we called such concept representation modulo satisfiability or representation in the $Ł_\infty$-MODSAT system. 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 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 to efficiently build representations in a logical framework whose associated problems have reasonable complexity and there is considerable literature about them.

Moreover, as an application of the aforementioned representational framework, we used the reasoning structure of Łukasiewicz Infinitely-valued Logic to set up a framework of formal analysis of reachability and robustness of neural networks, which are functions that may be approximated by representations in $Ł_\infty$-MODSAT. We also presented the results of our analysis of an actual neural network that is an exact rational McNaughton function.

In the area of probabilities, we provided a theoretical basis for the development and implementation of probabilistic reasoning over Łukasiewicz Infinitely-valued Logic, specifically to solve the ŁIPSAT problem. Analogous to the PSAT-solving of Finger and De Bona (2015), semantics modulo satisfiability played an important role in ŁIPSAT-solving due to the input atomic normal form, which states probabilistic assignments to propositional variables intending to be in accordance with a propositional theory. A phase transition behavior was empirically observed.

In the last subject we dealt with, we addressed a scenario in game theory which is unlike the ones in the previous applications since, instead of taking place a priori in a logical framework, it was translated into one in order to achieve the means to solve the raised problems. The problem

of PCE-Coherence was reduced either to PSAT or ŁIPSAT in a way that heavily subsumes semantics modulo satisfiability for assigning probabilities agreeing with a propositional theory; again, this shows how natural semantics modulo satisfiability are. Such reductions have made it possible to provide complexity analysis and algorithms for PCE-Coherence through both PSAT- and ŁIPSAT-solving. PCE-Coherence over GNP-classes were shown to be in NP and to be NP-complete over NP-complete GNP-classes. We also provided a reduction from PCE-Extension to PCE-Coherence.

6.1 Future Work

For the future, other applications of semantics modulo satisfiability may be identified in a variety of logical systems both to approach established problems — as in the cases of ŁIPSAT and PCE-Coherence — and define new useful concepts — as in the case of $Ł_\infty$-MODSAT. For instance, the work of Finger (2019) has rephrased decision problems as CPL-SAT and PSAT in a way that only admits a restricted underlying semantics; these problems turned out to be polynomial. The rephrased problems have not been placed in the framework of semantics modulo satisfiability, however, we believe such approach would perfectly fit them. In addition, the applications presented in this thesis also have room for further study.

Both the algorithm for building representations in $Ł_\infty$-MODSAT and its implementation might be improved in order to achieve efficiency gains. The formal analysis of neural networks is a work still in its dawn in which we may envision two directions to pursue: the effective codification of general neural networks as (approximate) rational McNaughton functions in the regional format encoding, so representable in $Ł_\infty$-MODSAT by our algorithm, and the formalization of more of their desirable properties in the language of Łukasiewicz Infinitely-valued Logic with a view to the emerging field of verification of neural networks. Despite our identification of neural networks with the functions that they determine, the experiment in Section 3.6.1 shows that it is not a trivial problem to translate from the neural network typical graph models to rational McNaughton functions in regional format. Also, despite the extensive literature on computational problems related to $Ł_\infty$, to effectively perform robustness verifications as proposed, an efficient approach to the decision on the validity of logical consequence in Łukasiewicz Infinitely-valued Logic is still needed, as the experiment also made clear.

Moreover, Amato *et al.* (2002) raised the hypothesis that the representation of neural networks in a logical system may be useful in their interpretability, which happens to be a challenge to their development; thus, approximate representations in $Ł_\infty$-MODSAT might play a role in such endeavor.

On the ŁIPSAT problem, one may focus on the improvement of the solvers having the analysis of the phase transition as a qualitative guideline. From a theoretical perspective, it may be worth investigating the connections between probabilities in accordance with a propositional theory $\mathrm{Th}(\Gamma)$, where a distribution only assigns positive probability to valuations in a semantics modulo satisfiability $\mathbf{Val}_\Gamma$, and conditional probabilities given that the formulas in Γ have occurred.

Finally, concerning observable games, besides tackling some practical problems and implementations of PCE-Coherence and PCE-Extension, the techniques presented might be expanded to other forms of equilibrium such as ε-Nash equilibrium (Daskalakis *et al.*, 2009).

Bibliography

Achterberg(2009) Tobias Achterberg. Scip: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41. See http://scip.zib.de/. Cited in p. 70

Aguzzoli(1998) Stefano Aguzzoli. The complexity of McNaughton functions of one variable. *Advances in Applied Mathematics*, 21(1):58–77. Cited in p. 56

Aguzzoli and Ciabattoni(2000) Stefano Aguzzoli and Agata Ciabattoni. Finiteness in infinite-valued Łukasiewicz logic. *Journal of Logic, Language and Information*, 9:5–29. doi: 10.1023/A:1008311022292. URL http://doi.org/10.1023/A:1008311022292. Cited in p. 52

Aguzzoli and Mundici(2001) Stefano Aguzzoli and Daniele Mundici. Weierstrass approximations by Łukasiewicz formulas with one quantified variable. In *Proceedings 31st IEEE International Symposium on Multiple-Valued Logic*, pages 361–366. IEEE. Cited in p. 15, 56

Aguzzoli and Mundici(2003) Stefano Aguzzoli and Daniele Mundici. Weierstrass approximation theorem and Łukasiewicz formulas with one quantified variable. In Melvin Fitting and Ewa Orłowska, editors, *Beyond Two: Theory and Applications of Multiple-Valued Logic*, pages 315–335. Physica-Verlag HD, Heidelberg. ISBN 978-3-7908-1769-0. Cited in p. 15, 56

Amato and Porto(2000) P. Amato and M. Porto. An algorithm for the automatic generation of a logical formula representing a control law. *Neural Network World*, 10(5):777–786. Cited in p. 15

Amato *et al.***(2002)** Paolo Amato, Antonio Di Nola and Brunella Gerla. Neural networks and rational Łukasiewicz logic. In *2002 Annual Meeting of the North American Fuzzy Information Processing Society Proceedings. NAFIPS-FLINT 2002 (Cat. No. 02TH8622)*, pages 506–510. IEEE. Cited in p. 98

Ansótegui *et al.***(2012)** C. Ansótegui, M. Bofill, F. Manyà and M. Villaret. Building automated theorem provers for infinitely-valued logics with satisfiability modulo theory solvers. In *2012 IEEE 42nd International Symposium on Multiple-Valued Logic*, pages 25–30. Cited in p. 42, 55

Aumann(1974) Robert J. Aumann. Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, 1(1):67–96. Cited in p. 77

Barrett *et al.***(2016)** Clark Barrett, Pascal Fontaine and Cesare Tinelli. The satisfiability modulo theories library (SMT-LIB). www.SMT-LIB.org, 2016. Cited in p. 42, 55

Bertsimas and Tsitsiklis(1997) Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to linear optimization*. Athena scientific series in optimization and neural computation. Athena Scientific, Belmont. ISBN 1886529191. Cited in p. 12

Bofill *et al.***(2015)** Miquel Bofill, Felip Manyà, Amanda Vidal and Mateu Villaret. Finding hard instances of satisfiability in Łukasiewicz logics. In *2015 IEEE International Symposium on Multiple-Valued Logic (ISMVL)*, pages 30–35. Cited in p. 2, 55, 59, 70

Boole(1854) George Boole. *An Investigation on the Laws of Thought*. Macmillan, London. Available on project Gutemberg at www.gutenberg.org/etext/15114. Cited in p. 10

Borgward(1986) Karl Heinz Borgward. *The Simplex Method: A Probabilistic Analysis*. Algorithms and Combinatorics 1. Springer. ISBN 978-3540170969. Cited in p. 13

Bova and Flaminio(2010) Simone Bova and Tommaso Flaminio. The coherence of Łukasiewicz assessments is NP-complete. *International Journal of Approximate Reasoning*, 51(3):294–304. ISSN 0888-613X. doi: 10.1016/j.ijar.2009.10.002. URL www.sciencedirect.com/science/article/pii/S0888613X09001558. Cited in p. 2, 59, 62, 63

Brøndsted(1983) Arne Brøndsted. *An Introduction to Convex Polytopes*, volume 90 of *Graduate Texts in Mathematics*. Springer-Verlag, New York. Cited in p. 11, 12

Cheeseman *et al.***(1991)** Peter Cheeseman, Bob Kanefsky and William M. Taylor. Where the really hard problems are. In *Proceedings of the 12th international joint conference on Artificial intelligence*, volume 1, pages 331–337, San Francisco. Morgan Kaufmann. ISBN 1-55860-160-0. URL http://portal.acm.org/citation.cfm?id=1631171.1631221. Cited in p. 13, 69, 70

Cignoli *et al.***(2000)** R.L. Cignoli, I.M. D'Ottaviano and D. Mundici. *Algebraic Foundations of Many-Valued Reasoning*. Trends in Logic. Springer Netherlands. ISBN 9789401594806. Cited in p. 9, 23, 24

Conitzer and Sandholm(2008) Vincent Conitzer and Tuomas Sandholm. New complexity results about Nash equilibria. *Games and Economic Behavior*, 63(2):621–641. Cited in p. 92, 93

Cook(1971) Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA. Association for Computing Machinery. ISBN 9781450374644. doi: 10.1145/800157.805047. URL http://doi.org/10.1145/800157.805047. Cited in p. 6, 9, 78

Daskalakis *et al.***(2009)** Constantinos Daskalakis, Paul W. Goldberg and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259. Cited in p. 98

de Finetti(1931) Bruno de Finetti. Sul significato soggettivo della probabilità. *Fundamenta Mathematicae*, 17(1):298–329. URL http://eudml.org/doc/212523. Translated into English as "On the Subjective Meaning of Probability", In: P. Monari and D. Cocchi (Eds.), Probabilità e Induzione, Clueb, Bologna, 291-321, 1993. Cited in p. 61, 74

de Finetti(1937) Bruno de Finetti. La prévision: Ses lois logiques, ses sources subjectives, 1937. Cited in p. 61, 74

de Finetti(2017) Bruno de Finetti. *Theory of probability: A critical introductory treatment. Translated by Antonio Machí and Adrian Smith*. John Wiley & Sons. Cited in p. 61, 74

Di Nola and Leuștean(2011) Antonio Di Nola and Ioana Leuștean. Riesz MV-algebras and their logic. In *Proceedings of the 7th conference of the European Society for Fuzzy Logic and Technology (EUSFLAT-11)*, pages 140–145. Atlantis Press. ISBN 978-90-78677-00-0. doi: 10.2991/eusflat.2011.125. Cited in p. 56

Di Nola and Leuștean(2014) Antonio Di Nola and Ioana Leuștean. Łukasiewicz logic and Riesz spaces. *Soft Computing*, 18:2349–2363. doi: 10.1007/s00500-014-1348-z. Cited in p. 56

Dutertre(2014) Bruno Dutertre. Yices 2.2. In Armin Biere and Roderick Bloem, editors, *Computer-Aided Verification (CAV'2014)*, volume 8559 of *Lecture Notes in Computer Science*, pages 737–744. Springer. Cited in p. 42, 55, 70

Enderton(2001) Herbert B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 2nd edition. ISBN 978-0122384523. Cited in p. 8

Esteva *et al.***(2001)** Francesc Esteva, Lluís Godo and Franco Montagna. The Π and $\Pi\frac{1}{2}$ logics: two complete fuzzy systems joining Łukasiewicz and product logics. *Archive for Mathematical Logic*, 40(1):39–67. ISSN 1432-0665. Cited in p. 55

Etessami and Yannakakis(2010) Kousha Etessami and Mihalis Yannakakis. On the complexity of Nash equilibria and other fixed points. *SIAM Journal on Computing*, 39(6):2531–2597. Cited in p. 87

Finger(2019) Marcelo Finger. Sparse models: a tractable fragment for SAT, MAXSAT and PSAT. In Cezar Augusto Mortari, Ricardo Silvestre, Ítala Maria Loffredo D'Ottaviano, Leandro Suguitani and Petrucio Viana, editors, *19th Brazilian Logic Conference EBL 2019: Book of Abstracts*, pages 128–129. Mídia Gráfica e Editora Ltda, UFCG-EDUFCG. Cited in p. 98

Finger(2020) Marcelo Finger. Logic in times of big data. In J. Acacio de Barros and Décio Krause, editors, *A True Polymath: A Tribute to Francisco Antonio Doria*, pages 184–198. College Publications. ISBN 978-1-84890-351-7. URL www.collegepublications.co.uk/ABF/?00002. Cited in p. 49

Finger and Bona(2011) Marcelo Finger and Glauber De Bona. Probabilistic satisfiability: Logic-based algorithms and phase transition. In *IJCAI*, pages 528–533. Cited in p. 11, 62, 66, 70

Finger and De Bona(2015) Marcelo Finger and Glauber De Bona. Probabilistic satisfiability: algorithms with the presence and absence of a phase transition. *Annals of Mathematics and Artificial Intelligence*, 75(3):351–379. ISSN 1012-2443. doi: 10.1007/s10472-015-9466-6. URL http://dx.doi.org/10.1007/s10472-015-9466-6. Cited in p. 2, 11, 12, 62, 66, 72, 97

Finger and Preto(2018) Marcelo Finger and Sandro Preto. Probably half true: Probabilistic satisfiability over Łukasiewicz infinitely-valued logic. In Didier Galmiche, Stephan Schulz and Roberto Sebastiani, editors, *Automated Reasoning. IJCAR 2018*, volume 10900 of *Lecture Notes in Computer Science*, pages 194–210, Cham. Springer International Publishing. ISBN 978-3-319-94205-6. Cited in p. 4, 72

Finger and Preto(2020) Marcelo Finger and Sandro Preto. Probably partially true: Satisfiability for Łukasiewicz infinitely-valued probabilistic logic and related topics. *Journal of Automated Reasoning*, 64(7):1269–1286. ISSN 1573-0670. doi: 10.1007/s10817-020-09558-9. URL http://doi.org/10.1007/s10817-020-09558-9. Cited in p. 3, 57, 72

Fischer *et al.***(2006)** Felix Fischer, Markus Holzer and Stefan Katzenbeisser. The influence of neighbourhood and choice on the complexity of finding pure Nash equilibria. *Information Processing Letters*, 99(6):239–245. Cited in p. 78

Gamrath *et al.***(2020)** Gerald Gamrath, Daniel Anderson, Ksenia Bestuzheva, Wei-Kun Chen, Leon Eifler, Maxime Gasse, Patrick Gemander, Ambros Gleixner, Leona Gottwald, Katrin Halbig, Gregor Hendel, Christopher Hojny, Thorsten Koch, Pierre Le Bodic, Stephen J. Maher, Frederic Matter, Matthias Miltenberger, Erik Mühmer, Benjamin Müller, Marc Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Christine Tawfik, Stefan Vigerske, Fabian Wegscheider, Dieter Weninger and Jakob Witzig. The SCIP optimization suite 7.0. Technical report, Optimization Online. URL www.optimization-online.org/DB_HTML/2020/03/7705.html. Cited in p. 42

Gent and Walsh(1994) Ian P. Gent and Toby Walsh. The SAT phase transition. In *ECAI94 – Proceedings of the Eleventh European Conference on Artificial Intelligence*, pages 105–109. John Wiley & Sons. Cited in p. 70

Georgakopoulos *et al.***(1988)** George Georgakopoulos, Dimitris Kavvadias and Christos H. Papadimitriou. Probabilistic satisfiability. *Journal of Complexity*, 4(1):1–11. ISSN 0885-064X. doi: 10.1016/0885-064X(88)90006-4. Cited in p. 11, 62

Gerla(2001) B. Gerla. Rational Łukasiewicz logic and DMV-algebras. *Neural Network World*, 11 (6):579–594. Cited in p. 56

Gilboa and Zemel(1989) Itzhak Gilboa and Eitan Zemel. Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior*, 1(1):80–93. Cited in p. 93

Goldreich(2008) Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, Cambridge. Cited in p. 13

Goodfellow *et al.***(2016)** Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning*. MIT Press. www.deeplearningbook.org. Cited in p. 49

Gottlob *et al.***(2005)** G. Gottlob, G. Greco and F. Scarcello. Pure Nash equilibria: Hard and easy games. *Journal of Artificial Intelligence Research*, 24:357–406. Cited in p. 78

Hähnle(1991) Reiner Hähnle. Towards an efficient tableau proof procedure for multiple-valued logics. In Egon Börger, Hans Kleine Büning, Michael M. Richter and Wolfgang Schönfeld, editors, *Computer Science Logic: 4th Workshop, CSL '90 Heidelberg, Germany, October 1–5, 1990 Proceedings*, pages 248–260. Springer, Berlin, Heidelberg. ISBN 978-3-540-38401-4. doi: 10.1007/3-540-54487-9_62. URL http://doi.org/10.1007/3-540-54487-9_62. Cited in p. 68

Hansen and Jaumard(2000) P. Hansen and B. Jaumard. Probabilistic satisfiability. In Dov M. Gabbay and Philippe Smets, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 5, pages 321–367. Springer Netherlands. Cited in p. 11

Hansen and Jaumard(1990) Pierre Hansen and Brigitte Jaumard. Algorithms for the maximum satisfiability problem. *Computing*, 44(4):279–303. URL http://dx.doi.org/10.1007/BF02241270. Cited in p. 65

Hughes and Anderson(1996) Robert B. Hughes and Michael R. Anderson. Simplexity of the cube. *Discrete Mathematics*, 158(1-3):99–150. Cited in p. 24

Kavvadias and Papadimitriou(1990) Dimitris Kavvadias and Christos H. Papadimitriou. A linear programming approach to reasoning about probabilities. *Annals of Mathematics and Artificial Intelligence*, 1(1):189–205. URL http://dx.doi.org/10.1007/BF01531078. Cited in p. 65

Leshno *et al.***(1993)** Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867. doi: 10.1016/S0893-6080(05)80131-5. URL www.sciencedirect.com/science/article/pii/S0893608005801315. Cited in p. 49

Levin(1973) Leonid Anatolevich Levin. Universal sequential search problems. *Problemy Peredachi Informatsii*, 9(3):115–116. Cited in p. 9

McFarland(1993) Michael C. McFarland. Formal verification of sequential hardware: A tutorial. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(5):633–654. Cited in p. 6

McNaughton(1951) R. McNaughton. A theorem about infinite-valued sentential logic. *Journal of Symbolic Logic*, 16:1–13. Cited in p. 17

Mundici(2011) D. Mundici. *Advanced Łukasiewicz calculus and MV-algebras*. Trends in Logic. Springer Netherlands. ISBN 9789400708402. Cited in p. 60

Mundici(1987) Daniele Mundici. Satisfiability in many-valued sentential logic is NP-complete. *Theoretical Computer Science*, 52(1-2):145–153. Cited in p. 2, 9, 50, 55, 63

Mundici(1994) Daniele Mundici. A constructive proof of McNaughton's theorem in infinite-valued logic. *The Journal of Symbolic Logic*, 59(2):596–602. Cited in p. 21, 23, 29

Mundici(2006) Daniele Mundici. Bookmaking over infinite-valued events. *International Journal of Approximate Reasoning*, 43(3):223–240. ISSN 0888-613X. doi: 10.1016/j.ijar.2006.04.004. URL www.sciencedirect.com/science/article/pii/S0888613X0600034X. Cited in p. 59, 62

Munkres(2000) James R. Munkres. *Topology*. Prentice Hall, Upper Saddle River. Cited in p. 12

Nash(1951) John Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295. ISSN 0003486X. URL www.jstor.org/stable/1969529. Cited in p. 73, 74, 86

Nash(1950a) John F. Nash. *Non-Cooperative Games*. PhD thesis, Princeton University. Cited in p. 74

Nash(1950b) John F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1):48–49. Cited in p. 74

Nilsson(1986) Nils Nilsson. Probabilistic logic. *Artificial Intelligence*, 28(1):71–87. Cited in p. 11, 62

Papadimitriou and Steiglitz(1998) C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover. Cited in p. 12

Papadimitriou(1994) Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, Boston. Cited in p. 13

Papadimitriou(2007) Christos H. Papadimitriou. The complexity of finding Nash equilibria. In Noam Nisan, Tim Roughgarden, Eva Tardos and Vijay V. Vazirani, editors, *Algorithmic game theory*, pages 29–51. Cambridge University Press. Cited in p. 87

Papadimitriou and Roughgarden(2008) Christos H. Papadimitriou and Tim Roughgarden. Computing correlated equilibria in multi-player games. *Journal of the ACM*, 55(3):1–29. ISSN 0004-5411. doi: 10.1145/1379759.1379762. URL http://doi.org/10.1145/1379759.1379762. Cited in p. 77, 94

Preto and Finger(2019) Sandro Preto and Marcelo Finger. Representing rational McNaughton functions via MODSAT relativisation. In Cezar Augusto Mortari, Ricardo Silvestre, Ítala Maria Loffredo D'Ottaviano, Leandro Suguitani and Petrucio Viana, editors, *19th Brazilian Logic Conference EBL 2019: Book of Abstracts*, page 183. Mídia Gráfica e Editora Ltda, UFCG-EDUFCG. Cited in p. 3, 57

Preto and Finger(2020) Sandro Preto and Marcelo Finger. An efficient algorithm for representing piecewise linear functions into logic. *Electronic Notes in Theoretical Computer Science*, 351:167–186. ISSN 1571-0661. doi: 10.1016/j.entcs.2020.08.009. URL http://doi.org/10.1016/j.entcs.2020.08.009. Proceedings of LSFA 2020, the 15th International Workshop on Logical and Semantic Frameworks, with Applications (LSFA 2020). Cited in p. 3, 57

Salvatore *et al.***(2019)** Felipe Salvatore, Sandro Preto, Marcelo Finger and Roberto Hirata Jr. Using neural models to perform inference. In Derek Doran, Artur d'Avila Garcez and Freddy Lecue, editors, *Proceedings of the 2019 International Workshop on Neural-Symbolic Learning and Reasoning*. Cited in p. 3

Szegedy *et al.***(2014)** Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Cited in p. 51

Tarela and Martínez(1999) J.M. Tarela and M.V. Martínez. Region configurations for realizability of lattice piecewise-linear models. *Mathematical and Computer Modelling*, 30(11-12):17–27. Cited in p. 40

Tarela *et al.***(1990)** J.M. Tarela, E. Alonso and M.V. Martínez. A representation method for PWL functions oriented to parallel processing. *Mathematical and Computer Modelling*, 13(10):75 – 83. ISSN 0895-7177. doi: 10.1016/0895-7177(90)90090-A. URL www.sciencedirect.com/science/article/pii/089571779090090A. Cited in p. 40

Xu and Wang(2019) J. Xu and S. Wang. Lattice piecewise affine representations on convex projection regions. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 7240–7245. doi: 10.1109/CDC40024.2019.9030119. Cited in p. 40