

Computable Analysis and Game Theory: From Foundations to Applications

Tonicha Crook

Submitted to Swansea University in fulfilment
of the requirements for the Degree of Doctor of Philosophy



Swansea University
Prifysgol Abertawe

Department of Computer Science
Swansea University

September 2023

Declarations

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed..... 

Date.....03/07/2024.....

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed..... 

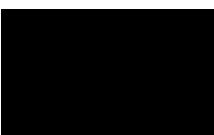
Date.....03/07/2024.....

I hereby give consent for my thesis, if accepted, to be available for electronic sharing

Signed..... 

Date.....03/07/2024.....

The University's ethical procedures have been followed and, where appropriate, that ethical approval has been granted.

Signed..... 

Date.....03/07/2024.....

Abstract

This body of research showcases several facets of the intersection between computer science and game theory.

On the foundational side, we explore the obstructions to the computability of Nash equilibria in the setting of computable analysis. In particular, we study the Weihrauch degree of the problem of finding a Nash equilibrium for a multiplayer game in normal form. We conclude that the Weihrauch degree Nash for multiplayer games lies between $\text{AoUC}_{[0,1]}^*$ and $\text{AoUC}_{[0,1]}^\circ$ (Theorem 5.3). As a slight detour, we also explore the demarcation between computable and non-computable computational problems pertaining to the verification of machine learning. We demonstrate that many verification questions are computable without the need to specify a machine learning framework (Section 7.2). As well as looking into the theory of learners, robustness and sparsity of training data.

On the application side, we study the use of Hypergames in Cybersecurity. We look into cybersecurity AND/OR attack graphs and how we could turn them into a hypergame (8.1). Hyper Nash equilibria is not an ideal solution for these games, however, we propose a regret-minimisation based solution concept. In Section 8.2, we survey the area of Hypergames and their connection to cybersecurity, showing that even if there is a small overlap, the reach is limited. We suggest new research directions such as adaptive games, generalisation and transferability (Section 8.3).

Acknowledgements

I would like to thank my supervisor, Arno Pauly, for introducing me to Computable Analysis and expanding my knowledge of game theory. I would also like to express my gratitude to my CDT AIMLAC¹, which provided the best PhD experience I could have asked for. They organised amazing cohort events/training, provided many exciting conference opportunities and expanded my horizons via an industrial placement. Specifically, I want to thank Roz Toft for her support during my first year and for always being there for me.

Next, I want to extend my thanks to my boyfriend, Liam, who put up with me and helped me with my work, encouraging me to leave the house and take breaks when it was getting stressful. I also want to thank my family for their unwavering support throughout this journey - my Mum, Dad, Sabrina, and Pete. Pete, I appreciate you reading through this thesis; your comments were consistently valuable. I want to express my gratitude to all my friends, who added fun and excitement to my life. Whether they were part of the CoFo or from outside of university, they were always there to cheer me on. Finally, I want to thank Jeremy for coming through at the last minute to drive me to my Viva on time!

¹This work is supported by the UKRI AIMLAC CDT, cdt-aimlac.org, grant no. EP/S023992/1.

Contents

I	Introduction	1
1	Introduction	3
1.1	Aims and Contributions	4
1.2	Published Works	6
1.3	Thesis Overview	7
II	Background	9
2	Computable Analysis and Weihrauch Degrees	11
2.1	Represented Spaces and Multivalued Functions	11
2.2	Topological Properties	13
2.3	Weihrauch Reducibility	15
2.4	Choice Principles	20
2.5	Reverse Mathematics Principles	22
2.6	Translating Between Spaces of Subsets	24
3	Game Theory and Hypergame Theory	27
3.1	Game Theory	27
3.2	Machine Learning and Game Theory	31
3.3	Game Theory and Complexity	33
3.4	Hypergame Theory	35
4	Computability and Game Theory	41
4.1	Computability for Strategic Games	41
4.2	Computability for Sequential Games	42
4.3	Constructivism in Game Theory and Bounded Rationality	43
4.4	Applications of Games to Computability Theory	44
III	Computable Analysis and Game Theory	47
5	The Non-computability of Nash Equilibria in Multiplayer Games	49
5.1	Introduction	49

5.2	Our Main Theorem	51
5.3	Roots of Polynomials	52
5.4	An Open Question and a Remark	70
5.5	Consequences of the Classification	72
IV Machine Learning and Cybersecurity Applications		75
6	Introduction	77
7	A Computable Analysis Perspective on Verified Machine Learning	79
7.1	Introduction	79
7.2	A Theory of Verified Machine Learning	80
7.3	Related Work	93
7.4	Conclusions	94
8	Hypergames and Cybersecurity	95
8.1	Hypergames and Attack Graphs	95
8.2	Applications to Cybersecurity	103
8.3	Suggested Research Directions	110
8.4	Related Research Areas	114
8.5	Conclusions	116
V Conclusions		119
9	Conclusions and Future Work	121
Bibliography		123
A	Robust/Sparse Code - Python	139
B	Hypergames and Hyper Nash Equilibrium - Python	143

Part I

Introduction

Chapter 1

Introduction

Contents

1.1	Aims and Contributions	4
1.2	Published Works	6
1.3	Thesis Overview	7

This work represents a fusion of disciplines, namely computable analysis, game theory and machine learning. Comprehending the foundations of these fields is essential to grasp the collaboration within this thesis. From this we explore the practical applications in the various domains, notably cybersecurity. This multidisciplinary approach demonstrates the power of interdisciplinary collaboration.

Computable analysis is a branch of mathematics that focuses on studying functions, numbers, and mathematical structures from a computational perspective. It aims to understand what can be effectively computed within the realm of real numbers and continuous mathematics. The groundwork for computable analysis was laid by Alan Turing’s development of the Turing machine, a theoretical model of computation [154]. Turing’s work in the 1930s established the concept of computability and paved the way for exploring which functions could be effectively computed by such machines. During the late 1930s and early 1940s, Turing and other notable figures, such as Alonzo Church and Stephen Kleene, contributed to the definition of computable numbers.

In this thesis, we delve into a specific area of computable analysis known as Weihrauch complexity. The concept of Weihrauch complexity has been present since the 1980s, with the foundational work by Klaus Weihrauch [162]. However, the precise definitions we currently employ were solidified only in 2008. A comprehensive database encompassing all papers related to Weihrauch complexity has been curated¹, consisting of over 160 papers. Notably, a significant portion of these papers has been published within the past five years. This trend highlights the recent emergence and increasing interest in this area.

¹<http://cca-net.de/publications/weibib.php>

Game theory received its initial comprehensive mathematical formulation from John von Neumann and Oskar Morgenstern [112]. Through their work, they introduced pivotal concepts such as zero-sum games, strategies, and the renowned minimax theorem. Subsequently, in the 1950s, John Nash introduced the concept of the Nash equilibrium [110], which signifies a state within a game where no player has an incentive to unilaterally deviate from their chosen strategy. Modern day game theory possesses a rich history that spans across diverse disciplines. Evolving from its foundational principles, it has developed into a versatile framework that aids in comprehending strategic decision-making across a wide spectrum of contexts.

A more recent subsection of game theory is Hypergame theory. hypergames are an intriguing concept that has emerged from the need to develop better models for games with incomplete information. The notion of hypergames involves each player behaving as an autonomous agent who makes decisions based on their individual ‘internal model’, which is constructed from their own knowledge of game states and their perception of other players’ knowledge of game states. This situation arises when a player does not know or fully understand all the strategies of the game.

Machine learning’s history spans from its conceptual roots in the mid-20th century, focusing on simulating human cognitive processes, through the rule-based systems of the 1970s and 1980s, neural network revivals in the 1990s, and the big data era of the early 2000s. The pivotal emergence of deep learning in the mid-2010s propelled machine learning into a new era, revolutionising tasks like image recognition and natural language processing. Now deeply integrated across industries, machine learning continues to evolve through advancements in neural architectures, reinforcement learning, and interdisciplinary collaborations with fields like robotics and economics.

This thesis represents a comprehensive culmination of my research endeavours, encompassing both my previously published works and those soon to be published. As a result, readers may notice that several chapters or sections have already been made available or are in the process of being released. It is essential to acknowledge that similarities may arise between these published or forthcoming works and the content within this thesis. However, I would like to highlight that extensive efforts have been invested in the refinement and alignment of the materials to ensure a coherent and harmonious structure throughout the thesis. The content has been meticulously rearranged and, where necessary, modified to enhance the overall flow and construct of the thesis. This approach allows the various pieces of research to converge seamlessly, contributing to a comprehensive and unified exploration of the subject matter.

1.1 Aims and Contributions

The first part of this project (Chapter 5) aims to use Weihrauch computability to explore game theory techniques and their level of computability. The exploration starts by building upon previous research conducted by Arno Pauly [124] regarding the Weihrauch degree of Nash equilibria. The research discovered that for one-player games, finding the Nash equilibrium corresponds to the Weihrauch degree of finite choice. For

two-player games, it was proven to be equivalent to All-or-Unique Choice ($\text{AoUC}_{[0,1]}^*$). Additionally, it was established by Pauly that Nash equilibrium for one-player games is strictly Weihrauch reducible to Nash equilibrium for two-player games. Expanding on this research involves demonstrating the incomputable nature of Nash equilibria in multiplayer games. While finding the Nash equilibrium in two-player games involves solving a single linear inequality, this complexity grows when extended to multiplayer games, resulting in higher-degree polynomial equations. We show that the Weihrauch degree of Nash equilibrium for multiplayer games can be compared to a benchmark principle $\text{AoUC}_{[0,1]}$. We proved that it lies between $\text{AoUC}_{[0,1]}$ invoked finitely many times in parallel and $\text{AoUC}_{[0,1]}$ invoked any finitely many times. Furthermore, we have established that Nash equilibrium is solvable with finitely many mind chances, is Las Vegas computable and Monte Carlo computable.

In the second project (Chapter 7), we harness computable analysis to delve into machine learning, exploring a new dimension in this field. By applying the principles of computable analysis, we seek to verify and validate classifiers and learners with a level of precision and rigour not previously attainable. This novel approach allows us to scrutinise the intricacies of machine learning algorithms, providing a deeper understanding of their behaviours and limitations. Through the lens of computable analysis, we provide results about which properties of classifiers and learners are computable. We explore various verification questions related to classifiers, with the goal of validating specific criteria and assessing the classifier’s performance over a designated set or region, which we proved to be computable. Additionally, we investigate the detection and prevention of adversarial examples by employing computable metric spaces and classifiers, incorporating the concept of ‘locallyConstant’. Furthermore, we delve into examining the robustness of learners that transform a finite sequence of labeled points into classifiers.

The third project (Chapter 8) leverages the framework of hypergames to explore the realm of cybersecurity, thus extending the boundaries of research in this domain. By employing hypergames, we embark on an exploration that goes beyond conventional methodologies, enabling us to dissect and comprehend the intricate dynamics of cybersecurity challenges. Through the lens of hypergames, we dissect the strategic interplay between attackers and defenders within cyber systems. This work combines a literature review paper which was a joint work with Andrew Fielder, Paul Jones, and Conor Artman, as well as some work converting cyber attack graphs to hypergames which was a joint work with Martin Barrere Cambrun and Chris Hankin. The conversion from cyber attack graphs to hypergames was relatively straightforward. However, we encountered challenges when finding the Hyper-Nash equilibrium. This was particularly complex because the two parties, attackers and defenders, were in competition. A higher payoff for the attacker indicates a successful breach of the system, resulting in a worse outcome for the defender and a lower payoff. Therefore, we suggest a different approach using regret minimisation, with joint work with Arno Pauly.

1.2 Published Works

The Weihrauch Degree of Finding Nash Equilibria in Multiplayer Games [49]

Presented in Chapter 5. Joint work with Arno Pauly.

Presents: Is there an algorithm that takes a game in normal form as input, and outputs a Nash equilibrium? If the payoffs are integers, the answer is yes, and a lot of work has been done in its computational complexity. If the payoffs are permitted to be real numbers, the answer is no, for continuity reasons. It is worthwhile to investigate the precise degree of non-computability (the Weihrauch degree) since knowing the degree entails what other approaches are available (eg, is there a randomised algorithm with positive success chance?). The two-player case has already been fully classified, but the multiplayer case remains open and is addressed here. Our approach involves classifying the degree of finding roots of polynomials and lifting this to systems of polynomial inequalities via cylindrical algebraic decomposition.

A Computability Perspective on (Verified) Machine Learning (WADT, [47])

Presented in Chapter 7. Joint work with Jay Morgan, Arno Pauly, Markus Roggenbach.

Presents: In Computer Science there is a strong consensus that it is highly desirable to combine the versatility of Machine Learning (ML) with the assurances formal verification can provide. However, it is unclear what such ‘verified ML’ should look like.

This paper is the first to formalise the concepts of classifiers and learners in ML in terms of Computable Analysis. It provides results about which properties of classifiers and learners are computable. By doing this we establish a bridge between the continuous mathematics underpinning ML and the discrete setting of most of computer science.

We define the computational tasks underlying the newly suggested verified ML in a model-agnostic way, and show that they are in principle computable. Our formalisations are justified and proofs of theorems are provided.

Exploring the Non-Computability of Machine Learning Classifiers (Extended Abstract)[48]

Joint work with Jay Morgan, Arno Pauly, Markus Roggenbach.

Presents: We have previously shown that various questions pertaining to the verification of classifiers are computable. Here, we use Weihrauch degrees to examine how non-computable certain problems become. Such an approach reveals whether relaxing our demands for the nature of computation permits us to answer more questions.

Hypergames, Cybersecurity and RL (Publication Pending) Tonicha Crook, Andrew Fielder, Paul Jones, Conor Artman.

Presents: hypergames model games with incomplete information between players who each have different perspectives of the game being played. This can mean something as simple as different preferences or as complex as players having hidden strategies. Game theory has been used to explore cybersecurity for many years, however, hypergames are a new addition. An insight into the definitions and solution concepts are explored. Thereafter, a literature review of the current applications of hypergames in cybersecurity.

We present potential areas hypergames could be useful in the future along with suggested research areas.

1.3 Thesis Overview

- Part 2 provides the background needed for this thesis. This gives introductions to Computable Analysis (Chapter 2) and Game Theory (Chapter 3). As well as exploring the intersection of Computable Analysis and Game Theory (Chapter 4).
- Part 3 involves classifying the Weihrauch degree of Nash Equilibria in Multiplayer Games in order to explore how non-computable the task is (Chapter 5). The prerequisites for this chapter is Chapter 2.
- Part 4 are the applied chapters of this thesis. First is Verified Machine Learning (Chapter 7) which requires the knowledge of Chapter 2. Secondly is hypergames and Cybersecurity (Chapter 8) which requires Chapter 3.
- Part 5 is the Conclusion (Chapter 9).

Part II

Background

Chapter 2

Computable Analysis and Weihrauch Degrees

Contents

2.1	Represented Spaces and Multivalued Functions	11
2.2	Topological Properties	13
2.3	Weihrauch Reducibility	15
2.4	Choice Principles	20
2.5	Reverse Mathematics Principles	22
2.6	Translating Between Spaces of Subsets	24

The following summarises the formal definitions and key properties of the most important notions from Computable Analysis for this thesis. Focusing on Weihrauch computability, where the Weihrauch lattice framework allows us to classify the uniform computational content of problems and theorems from analysis and other areas of mathematics. A standard textbook is [164]. A quick introduction in a similar style is available as [25]. A concise, more general treatment is found in [126].

2.1 Represented Spaces and Multivalued Functions

In this section, our objective is to delve into the computability properties and requisite data types for our problems represented as $f : \subseteq X \rightrightarrows Y$. To achieve this, we establish a structure on the spaces X and Y through the concept of representations.

Definition 2.1 (Represented Spaces) A represented space \mathbf{X} is a pair (X, δ) which is a set X together with a surjective partial function $\delta : \subseteq \mathbb{N}^{\mathbb{N}} \rightarrow X$.

The term ‘representation’ refers to the map δ itself; when $\delta(p) = x$, we label p as a ‘name’ for x . For brevity, we may refer to a represented space as simply X if the representation is evident from the context or does not require explicit mention. We

borrow this concept from [30] and it is more extensively elaborated upon in [126]. In the context of represented spaces X and Y , we define problems as partial multivalued functions, denoted as $f : \subseteq X \rightrightarrows Y$. Properties like computability and continuity of problems can be readily introduced through the concept of realisers.

Definition 2.2 (Realiser) Given represented spaces $(X, \delta_X), (Y, \delta_Y)$, a problem $f : \subseteq X \rightrightarrows Y$ and a function $F : \subseteq \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}^{\mathbb{N}}$, we define $F \vdash f : \iff \delta_Y F \sqsubseteq f \delta_X$. We say that F is a realiser of f .

In essence, F is recognised as a realiser of f when $\delta_Y F$ effectively solves the problem $f \delta_X$, this depends on the underlying represented spaces. The validity of this notation, $F \vdash f$, hinges on the clarity of the underlying represented spaces. For Baire space, the continuity of functions is evident as $F : \subseteq \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}^{\mathbb{N}}$, while their computability can be formally defined using Turing machines. Consequently, we classify a problem f as ‘computable’ when it possesses a computable realiser. Likewise, if a continuous realiser can be found for f , it is termed ‘continuous’.

Interestingly, numerous spaces encountered in analysis can be classified as computable metric spaces. In this context, we work under the assumption that the reader is familiar with the concept of a computable (double) sequence of real numbers.

Definition 2.3 (Computable Metric Spaces) A computable metric space (X, d, α) is a separable metric space (X, d) with metric $d : X \times X \rightarrow \mathbb{R}$ and a dense sequence $\alpha : \mathbb{N} \rightarrow X$ such that $d \circ (\alpha \times \alpha) : \mathbb{N}^2 \rightarrow \mathbb{R}$ is a computable double sequence of real numbers.

A comprehensive introduction to multivalued functions within computable analysis can be found in [125]. The category-theoretic framework is established based on the composition of multivalued functions. Within this framework, they demonstrate that many-one degrees of multivalued functions form a distributive lattice.

We represent mathematical problems using partial multivalued functions, denoted as $f : \subseteq X \rightrightarrows Y$, where these functions are essentially relations $f \subseteq X \times Y$. For the purpose of our discussion, we define $\text{dom}(f) = \{x \in X : f(x) \neq \emptyset\}$ as the set of admissible instances for the problem. An admissible instance refers to a specific input value for a mathematical problem that can be effectively processed or analysed within the framework of computability. The corresponding set of function values, denoted as $f(x) \subseteq Y$, represents the set of possible outcomes. In the case of a single-valued function f , we treat $f(x)$ as a singleton value.

Mathematical problems can be combined in various ways to create new problems. In this thesis, we employ several standard operations to facilitate problem combinations. The subsequent definition outlines these operations, which are integral to our analysis. Given two represented spaces \mathbf{X}, \mathbf{Y} we obtain a third represented space $\mathcal{C}(\mathbf{X}, \mathbf{Y})$ of functions from X to Y by letting $0^n 1 p$ be a $\delta_X \rightarrow \delta_Y$ -name for f , if the n -th Turing machine equipped with the oracle p computes a realiser for f .

Definition 2.4 ([127]) Let $\mathbf{X} = (X, \delta_X), \mathbf{Y} = (Y, \delta_Y), \mathbf{Z} = (Z, \delta_Z), \mathbf{U} = (U, \delta_U)$ be

represented spaces.

1. Composition: $\mathcal{C}(\mathbf{Y}, \mathbf{Z}) \times \mathcal{C}(\mathbf{X}, \mathbf{Y}) \rightarrow \mathcal{C}(\mathbf{X}, \mathbf{Z})$.
2. Product: $\mathcal{C}(\mathbf{X}, \mathbf{Y}) \times \mathcal{C}(\mathbf{U}, \mathbf{Z}) \rightarrow \mathcal{C}(\mathbf{X} \times \mathbf{U}, \mathbf{Y} \times \mathbf{Z})$.
3. $\mathbf{X} \wedge \mathbf{Y} := (X \cap Y, \delta_X \wedge \delta_Y)$ where $(X \cap Y, \delta_X \wedge \delta_Y)(\langle p, q \rangle) = x$ iff $\delta_X(p) = x \wedge \delta_Y(p) = x$.

When considering two problems, denoted as f and g , we aim to define the concept of f solving g . The notation $f \sqsubseteq g$ indicates that every instance of problem g is also an instance of problem f , and furthermore, the solutions that are valid for both f and g correspond to valid solutions of problem g . These relationships exhibit both reflexivity and transitivity.

Definition 2.5 (Solutions) Let $f, g : \subseteq X \rightrightarrows Y$ be multivalued functions. We define $f \sqsubseteq g : \iff \text{dom}(g) \subseteq \text{dom}(f)$ and $(\forall x \in \text{dom}(g)) f(x) \subseteq g(x)$. In this situation we say that f solves g , f is a strengthening of g and g is a weakening of f .

2.2 Topological Properties

Within this thesis, we frequently regard spaces such as \mathbb{N} , \mathbb{R} , $[0, 1]$, $2^{\mathbb{N}}$, and $\mathbb{N}^{\mathbb{N}}$ as computable metric spaces. We make use of $\mathbb{N} = (\mathbb{N}, \delta_{\mathbb{N}})$ which is given by $\delta_{\mathbb{N}}(0^n 10^{\mathbb{N}}) = n$. On occasions where we require a non-metrisable space, we turn to the Sierpiński Space denoted as $\mathbb{S} = \{\perp, \top\}$.

Definition 2.6 (Sierpiński Space) Let $\mathbb{S} = (\{\perp, \top\}, \delta_{\mathbb{S}})$ be defined via $\delta_{\mathbb{S}}(0^{\mathbb{N}}) = 1$ and $\delta_{\mathbb{S}}(p) = 0$ for $p \neq 0^{\mathbb{N}}$.

Given a computable metric space (X, d, α) we denote by $B(x, r) := \{y \in X : d(x, y) < r\}$ the open ball with center $x \in X$ and radius $r \geq 0$.

Definition 2.7 (Open and Closed Sets) Denote open sets of a topological set X as $\mathcal{O}(\mathbf{X})$ if for every $x \in U$, there exists an $r > 0$ such that $B(x, r) \subseteq U$. Denote closed sets of a topological set X as $\mathcal{A}(\mathbf{X})$ by identifying $A \subseteq X$ with $(X \setminus A) \in \mathcal{O}(\mathbf{X})$.

Definition 2.8 (The Space of Open and Closed Sets) For any represented space \mathbf{X} we obtain two spaces of subsets of \mathbf{X} ; the space of open sets $\mathcal{O}(\mathbf{X})$ by identifying $f \in \mathcal{C}(\mathbf{X}, \mathbb{S})$ with $f^{-1}(\{\top\})$, and the space of closed sets \mathcal{A} by identifying $f \in \mathcal{C}(\mathbf{X}, \mathbb{S})$ with $f^{-1}(\{\perp\})$.

Definition 2.9 ([127]) Let \mathbf{X}, \mathbf{Y} be represented spaces then the following functions are well defined and computable:

1. Complement: $\mathcal{C} : \mathcal{O}(\mathbf{X}) \rightarrow \mathcal{A}(\mathbf{X}), \mathcal{C} : \mathcal{A}(\mathbf{X}) \rightarrow \mathcal{O}(\mathbf{X})$.
2. $\cup : \mathcal{O}(\mathbf{X}) \times \mathcal{O}(\mathbf{X}) \rightarrow \mathcal{O}(\mathbf{X}), \cup : \mathcal{A}(\mathbf{X}) \times \mathcal{A}(\mathbf{X}) \rightarrow \mathcal{A}(\mathbf{X})$.

3. $\cap : \mathcal{O}(\mathbf{X}) \times \mathcal{O}(\mathbf{X}) \rightarrow \mathcal{O}(\mathbf{X}), \cap : \mathcal{A}(\mathbf{X}) \times \mathcal{A}(\mathbf{X}) \rightarrow \mathcal{A}(\mathbf{X})$.
4. Union: $\bigcup : \mathcal{X}(\mathbf{N}, \mathcal{O}(\mathbf{X})) \rightarrow \mathcal{O}(\mathbf{X})$ maps a sequence $(U_n)_{n \in \mathbf{N}}$ of open sets to their union $\bigcup_{n \in \mathbf{N}} U_n$.
5. Intersection: $\bigcap : \mathcal{X}(\mathbf{N}, \mathcal{A}(\mathbf{X})) \rightarrow \mathcal{A}(\mathbf{X})$ maps a sequence $(A_n)_{n \in \mathbf{N}}$ of open sets to their union $\bigcap_{n \in \mathbf{N}} A_n$.
6. Product: $\times : \mathcal{O}(\mathbf{X}) \times \mathcal{O}(\mathbf{Y}) \rightarrow \mathcal{O}(\mathbf{X} \times \mathbf{Y}), \times : \mathcal{A}(\mathbf{X}) \times \mathcal{A}(\mathbf{Y}) \rightarrow \mathcal{A}(\mathbf{X} \times \mathbf{Y})$.

In addition to the concept of closed subsets, we also require the concept of a compact subsets.

Definition 2.10 (Computably Compact [127]) A represented space \mathbf{X} is (computably) compact, if the map $\text{IsEmpty}_{\mathbf{X}} : \mathcal{A}(\mathbf{X}) \rightarrow \mathcal{S}$ defined by $\text{IsEmpty}_{\mathbf{X}}(\emptyset) = \top$ and $\text{IsEmpty}_{\mathbf{X}}(A) = \perp$ otherwise is continuous (computable).

Definition 2.11 (Compact Sets) Define $\mathcal{K}(\mathbf{X})$ by identifying $K \subseteq X$ with $\{U \in \mathcal{O}(\mathbf{X}) \mid K \subseteq U\} \in \mathcal{O}(\mathcal{O}(\mathbf{X}))$ whenever $K = \bigcap \{U \in \mathcal{O}(\mathbf{X}) \mid K \subseteq U\}$.

Another perspective on this matter is that a representative space is computably compact if the map that is recognising that an open set is actually full is answering top and any other open set to bottom is computable. Additionally, a notable dual concept to compactness is overtness.

Definition 2.12 Let $\text{IsNonEmpty}_{\mathbf{X}} : \mathcal{O}(\mathbf{X}) \rightarrow \mathcal{S}$ be defined by $\text{IsNonEmpty}_{\mathbf{X}}(\emptyset) = \perp$ and $\text{IsNonEmpty}_{\mathbf{X}}(U) = \top$ for $U \neq \emptyset$. Now call \mathbf{X} (computably) overt, iff $\text{IsNonEmpty}_{\mathbf{X}}$ is continuous (computable).

Definition 2.13 (Overt Sets) Define $\mathcal{V}(\mathbf{X})$ by identifying $A \subseteq X$ with $\{U \in \mathcal{O}(\mathbf{X}) \mid A \cap U \neq \emptyset\} \in \mathcal{O}(\mathcal{O}(\mathbf{X}))$ whenever $A = \overline{A}$.

Overtness arises when the goal is to identify that an open set is non-empty. In this context, the response should be ‘yes’ if the input open set is not the empty set. This notion is dual to compactness, as compact subsets contain precisely the right amount of information to recognise inclusion in open sets. Conversely, overt subsets carry just enough information to identify intersections with open sets.

The relevance of $\mathcal{K}(\mathbf{X})$ and $\mathcal{V}(\mathbf{X})$ is found, in particular, in the following characterisations, which show that compactness just makes universal quantification preserve open predicates, and dually, overtness makes existential quantification preserve open predicates.

Proposition 2.14 ([126, Proposition 40 & 42]). The following are computable:

1. The map $\exists : \mathcal{O}(\mathbf{X} \times \mathbf{Y}) \times \mathcal{V}(\mathbf{X}) \rightarrow \mathcal{O}(\mathbf{Y})$ defined by

$$\exists(R, A) = \{y \in Y \mid \exists x \in A (x, y) \in R\}.$$

2. The map $\forall : \mathcal{O}(\mathbf{X} \times \mathbf{Y}) \times \mathcal{K}(\mathbf{X}) \rightarrow \mathcal{O}(\mathbf{Y})$ defined by

$$\forall(R, A) = \{y \in Y \mid \forall x \in A (x, y) \in R\}.$$

The represented space $(\mathcal{V} \wedge \mathcal{K})(\mathbf{X})$ contains the sets which are both compact and overt, and codes them by providing the compact and the overt information simultaneously. Thus, both universal and existential quantification over elements of $(\mathcal{V} \wedge \mathcal{K})(\mathbf{X})$ preserve open predicates.

A nice class of topological spaces, not necessarily based on countable sets, is formed by the category of CoPolish spaces. They play a significant role in Type-2 Complexity Theory [141] by simplifying complexity. CoPolish Spaces take a slightly different direction and can be understood as the direct limit of an increasing sequence of compact metrisable subspaces X_n . This concept is particularly significant in descriptive set theory and topology, providing a framework to explore topological and computational properties in a specific setting.

In the realm of topology, a topological space is deemed Hausdorff if, for any two distinct points x and y within the space, there exist open sets U and V that accommodate x and y respectively, while ensuring that these open sets do not intersect ($U \cap V = \emptyset$). This property, often referred to as the ‘Hausdorff separation axiom’, guarantees that any pair of distinct points can be separated by open sets.

2.3 Weihrauch Reducibility

Weihrauch computability was introduced by Klaus Weihrauch in the 1980s. The notion of Weihrauch reducibility was popularised by Brattka and Gherardi [22, 21]. There is a handbook that is an excellent introduction to the area [24]. The handbook begins by covering the basics of algebraic operations on multivalued functions and then introduces the theory of representations and computable functions of represented spaces. It’s important to understand both of these concepts before delving into Weihrauch computability, which is detailed in the next subsection. The handbook also discusses the algebraic and topological properties, completeness, composition, and implications of the Weihrauch lattice. Finally, it explores the choice problems and classifies different theorems from the analysis. An extension of this handbook and a collection of open questions in Weihrauch complexity can be found in [129].

Our aim now is to introduce Weihrauch reducibility as a means to compare problems to each other. The intent is for $f \leq_W g$ to convey the idea that problem f can be computed by a single application of problem g . The notion of computable reducibility for multivalued functions was introduced by Gherardi and Marcone within [63]. One problem is reducible to another, provided that whenever we have a method to compute a solution for the second problem, we can uniformly find a way to compute a solution for the first one. They show that the operator is transitive and reflexive.

To establish this concept, we require two variants of such a reducibility. We denote the identity of Baire space by $\text{id} : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}^{\mathbb{N}}$. For other sets X , we commonly include an index

and express the identity as $\text{id}_X : X \rightarrow X$. Given sets of functions $F, G : \subseteq \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}^{\mathbb{N}}$, we define $\langle F, G \rangle(p) := \langle F(p), G(p) \rangle$.

Definition 2.15 (Weihrauch Reducibility) Let f and g be problems. We define:

1. $f \leq_W g \iff (\exists \text{computable } H, K : \subseteq \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}^{\mathbb{N}} (\forall G \vdash g) H \langle \text{id}_{\mathbb{N}^{\mathbb{N}}}, GK \rangle \vdash f)$.
2. $f \leq_{sW} g \iff (\exists \text{computable } H, K : \subseteq \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}^{\mathbb{N}} (\forall G \vdash g) HGK \vdash f)$.

We say that f is (strongly) Weihrauch reducible to g if $f \leq_W g$ ($f \leq_{sW} g$) holds.

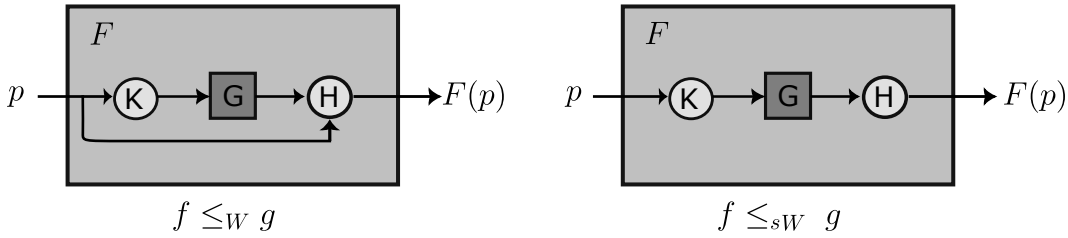


Figure 2.1: Weihrauch Reducibility and Strong Weihrauch Reducibility

The diagram in Figure 2.1 provides an illustration of Weihrauch reducibility and its stronger counterpart. It's evident that strong Weihrauch reducibility implies Weihrauch reducibility. Furthermore, both Weihrauch reducibility and strong Weihrauch reducibility satisfy the properties of preorders; that is, they are reflexive and transitive relations.

We denote the corresponding equivalences as \equiv_W and \equiv_{sW} , respectively. To signify strict reducibilities, we employ the symbols $<_W$ and $<_{sW}$. It's important to emphasise that the notation $f \leq_W g$ signifies that the oracle g is utilised once during the computation of f . This implies that utilising the oracle g could potentially present an obstacle, especially if the domain of g consists solely of intricate or complex points.

The equivalence classes established by \equiv_W and \equiv_{sW} are referred to as Weihrauch degrees and strong Weihrauch degrees, respectively. The reducibilities \leq_W and \leq_{sW} naturally extend to these degrees.

Definition 2.4 presents a set of algebraic operations that, when applied to Weihrauch degrees, offer an intuitive interpretation. Notably, while some operations have been excluded from this definition, several new operations have been introduced to enrich the framework.

Definition 2.16 (Operations on Weihrauch degrees) Let f, g be problems;

1. $f \sqcap g$ returns either an answer to f or an answer to g ,
2. $f \sqcup g$ lets us choose if f or g gives an answer,
3. $f \times g$ gets the answers to both f and g in parallel,

4. $f \star g$ lets us first apply g , then do some computation, and then apply f .
5. $f \rightarrow g = \min\{h \mid g \leq_W f \star h\}$,
6. f^* lets us invoke f finitely many times in parallel, meaning that all queries to f can be computed without knowing any of the answers.
7. f^\diamond lets us invoke f any finite number of times (not specified in advance), where later queries can be computed from previous answers [165].
8. \hat{f} lets us use f countably many times in parallel.
9. \bigsqcup where $\bigsqcup_{n \in \mathbb{N}} f_n$ receives an $n \in \mathbb{N}$ together with an input for f_n , and returns a matching output.

An integral map within the Weihrauch lattice is the limit map, which holds particular significance. When considering a Hausdorff space X , we establish the limit map (of Baire space) for this space. The domain of \lim_X consists of all converging sequences in X .

- Definition 2.17**
1. $\lim_X : \subseteq X^{\mathbb{N}} \rightarrow X, (x_n)_{n \in \mathbb{N}} \mapsto \lim_{n \rightarrow \infty} x_n$.
 2. $\lim : \subseteq \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}^{\mathbb{N}}, \langle p_0, p_1, p_2, \dots \rangle \mapsto \lim_{n \rightarrow \infty} p_n$

Brattka and Gherardi studied the Weihrauch reducibility of multivalued functions on represented spaces in [22]. Limited Principle of Omniscience (LPO) and Lesser Limited Principle of Omniscience (LLPO) have been introduced in constructive mathematics. LPO corresponds to the law of the excluded middle ($A \vee \neg A$) and LLPO to be de Morgan's law $\neg(A \vee B) \leftrightarrow (\neg A \vee \neg B)$. Where both are restricted to simple existential statements.

Definition 2.18 ([22]) We define:

1. $\text{LPO} : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}^{\mathbb{N}}, \text{LPO}(p) = \begin{cases} 0 & \text{if } (\exists n \in \mathbb{N}) p(n) = 0 \\ 1 & \text{otherwise} \end{cases}$
2. $\text{LLPO} : \subseteq \mathbb{N}^{\mathbb{N}} \rightrightarrows \mathbb{N}^{\mathbb{N}}, \text{LLPO}(p) \ni \begin{cases} 0 & \text{if } (\forall n \in \mathbb{N}) p(2n) = 0 \\ 1 & \text{if } (\forall n \in \mathbb{N}) p(2n + 1) = 0 \end{cases}$

where $\text{dom}(\text{LLPO}) := \{p \in \mathbb{N}^{\mathbb{N}} : p(k) \neq 0 \text{ for at most one } k\}$.

2.3.1 Structure of the Weihrauch Lattice

Higuchi and Pauly, [72], demonstrated that Weihrauch degrees and pointed Weihrauch degrees do not form a Brouwer algebra. They also explored the identification of other lattices that exhibit the structure of a Heyting or Brouwer algebra. For instance, while the computable Weihrauch lattice does not conform to a Heyting algebra, the continuous Weihrauch lattice does. While there are a few cases that remained open at the time of

this paper, the lattice allows finitely many parallel iterations of non-empty continuous Weihrauch degrees and the infinite version. Both of these are open concerning Brouwer algebra, however, the finite version has been confirmed to be true for Heyting algebra.

The investigation into Weihrauch degrees and their structure continues in [29] following the conclusion that they are not a Brouwer algebra. Instead, they look into the connections between intuitionistic linear logic and substructural logic. By employing reduction witnesses as inputs and outputs of multivalued functions they introduce two new operations; the compositional product of Weihrauch degrees and the implication.

The Weihrauch lattice serves as a tool for classifying the level of incomputability of mathematical problems. A selection of operators within the Weihrauch lattice are listed in [64]. This paper operates within the framework of the Type-2 Theory of Effectivity which provides a realistic and flexible model of computation. It defines negative, positive, and total closed projection operators, along with their approximated versions. Gherardi, Marcone and Pauly classify these operators using Weihrauch reducibility depending on the representation of closed sets and the dimension of the space. Depending on these factors they found that the projection and approximate projection operators characterise some of the most fundamental computational classes in the lattice.

The Cantor-Bendixson theorem utilises Polish spaces, perfect and countable sets. A new study into the Weihrauch lattice introduces the level of $\Pi_1^1 - \text{CA}_0$ and explores problems such as perfect subsets of Polish spaces, studying the perfect set theorem, the Cantor-Bendixson theorem [43]. They explore the connection of perfect sets and Cantor-Bendixson theorems in arbitrary computable metric spaces and achieve this by formulating the Cantor-Bendixson theorem as a problem using perfect kernels and scattered lists.

2.3.2 Related Reducibilities

In [12], Bauer introduces the concept of extended Weihrauch degrees. Extended Weihrauch degrees form a class derived from a preorder of Weihrauch reducibility, where the symmetrisation \equiv_W establishes an equivalence relation. Bauer extends this concept to encompass both modest extended Weihrauch predicates and $\neg\neg$ -dense extended Weihrauch predicates. Subsequently, they delve into examples of non-trivial non- $\neg\neg$ -degrees, the embedding of truth values, and reductions to and from non-moderate degrees.

In their work [55], Day, Downey, and Westrick employ Weihrauch reducibility in conjunction with parallelised Weihrauch reducibility to investigate discontinuous functions. In this context, they define three related reducibilities: \leq_T , \leq_{tt} , and \leq_m for arbitrary functions. They also examine the α -jump functions, highlighting their \leq_m -minimality within the Baire class and characterising the degree structures associated with \leq_T and \leq_{tt} within Baire 1 functions. These characterisations enable them to establish an exact match with the α hierarchy.

2.3.3 Some Applications of Weihrauch Computability

2.3.3.1 Supergraphs

In their paper [44], Cipriani and Pauly delve into the complexity of finding supergraphs. They examine various problem variations concerning fixed countable graphs, considering both subgraphs and induced subgraphs using the framework of Weihrauch reducibility and effective Wadge degrees. They address the question of whether there exist graphs for which determining their corresponding supergraph is more intricate. This question was initially posed in [13], where it was observed that for totally disconnected infinite graphs, deciding whether they are supergraphs of given graphs is equivalent to LPO, the simplest discontinuous problem. Cipriani and Pauly confirm the existence of a graph G such that $\text{LPO} <_W \text{IS}^G$ while also exploring the complexity of the search problem. They conclude that supergraph results do not appear to have a close relationship with analogous subgraph results. However, they do not currently have conjectures regarding the expected Weihrauch degrees in this context.

2.3.3.2 Gödel numbers

In [18], Brattka delves into the computability of Gödel numbers and demonstrates that the Gödel problem and its variants can be naturally classified within the Weihrauch lattice. They establish an optimal upper bound in terms of the Weihrauch version of the Kirby-Paris hierarchy, as well as investigate closure properties and lower bounds. Brattka confirms that $C_{\mathbb{N}}^{(n)}$ and $K_{\mathbb{N}}^{(n)}$ indeed correspond to $I\Omega_{n+1}^0$ and $B\Omega_{n+1}^0$, respectively. This classification allows for the categorisation of Gödel problems and their variants with respect to an appropriate benchmark scale.

Algorithmic learning theory investigates the problem of determining the Gödel number of a program given a computable sequence of natural numbers. Brattka's work [19] delves into this problem when dealing with computable sequences and classifies its Weihrauch complexity. In their study, they employ learning theory, closed and compact choice, and their jumps on natural numbers. By utilising the Weihrauch version of the Kirby-Paris hierarchy, Brattka demonstrates that the upper bound of G_{\geq} can be reduced to LPO^* and establishes that these bounds are minimal. Furthermore, they explore the closure properties of G , effective discontinuity, and lower bounds for Kolmogorov complexity.

2.3.3.3 Ramsey's Theorem

Dorais et al. showed that Ramsey's theorem for n -tuples and k many colours is not uniformly, or Weihrauch, reducible to Ramsey's theorem for n -tuples and j many colours [56]. They introduced the Squashing Theorem, enabling the deduction of multiple applications of a given principle which, in many cases, cannot be uniformly reduced to one. Their work also delves into Weihrauch reductions for thin sets, WWKL and the Rainbow Ramsey Theorem.

Pradic and Soldà explored the additive Ramsey theorem, which involves coloured pairs of rational numbers and incorporates additional colour-related structure [130]. They also introduced a simpler statement known as the shuffle principle. This principle asserts that every \mathcal{Q} -indexed word contains a convex subword in which each letter appears densely or not at all. Additionally, they employed Weihrauch complexity to determine the strength of an additive Ramseyan theorem over the rationals within the context of reverse mathematics. This determination was found to be Weihrauch equivalent to $\text{TC}_{\mathbb{N}}^* \times (\text{LPO}')^*$. Furthermore, it was established as equivalent to Σ_2^0 -induction and showed that the problem decomposes nicely into the distinct complexities $(\text{LPO}')^*$ or $\text{TC}_{\mathbb{N}}^*$.

2.3.3.4 Brouwer's Fixed Point Theorem

Brouwer's Fixed Point Theorem is a pivotal result stating that every continuous function $F : D \mapsto D$ mapping a compact convex set $D \subseteq \mathbb{R}^m$ to itself has a fixed point $x^* \in D$, such that $F(x^*) = x^*$. In other words, there exists a point within the set that remains unchanged under the function's transformation. These principles collectively form the backdrop for exploring the logical foundations of various mathematical phenomena.

The Brouwer Fixed Point Theorem was proved to be computably equivalent to connected to choice for any fixed dimension [27]. This equivalence is achieved by representing closed sets by trees of rational complexes. Both the Brouwer Fixed Point Theorem of dimension one and the Intermediate Value Theorem were found to not be idempotent, using a displacement principle that provides information on the power of binary choice on the left-hand side of a reduction.

2.4 Choice Principles

The Weihrauch lattice encompasses various Weihrauch degrees, including omniscience principles and choice principles. Choice principles represent multivalued functions with a non-empty, closed subset as an input and an element of the closed set as an output. Within [21], a range of choice principles are defined and examined as boundedness principles. Various different analyses and functional theorems are explored in order to classify their Weihrauch degree. Different techniques are used to achieve this, such as the parallelisation principle, the mind change principle and the Baire category principle.

The choice problem, denoted as C_X for a given space X , involves the task of locating a point within a specified closed set $A \subseteq X$. This seemingly simple yet fundamental problem leads to the derivation of various significant Weihrauch degrees by strategically selecting appropriate spaces X . Closed choice principles are multivalued functions taking as input a non-empty closed subset of some fixed space, and have to provide some element of the closed set as output.

Definition 2.19 (Closed Choice) Let \mathbf{X} be a represented space. Then the closed choice operation $C_{\mathbf{X}} : \subseteq \mathcal{A}(\mathbf{X}) \rightrightarrows \mathbf{X}$ of this space is defined by $x \in C_{\mathbf{X}}(A)$ iff $x \in A$.

Definition 2.20 Finite closed choice, denoted as C_k , operates within the ambient space $\{0, 1, \dots, k - 1\}$. The input consists of an enumeration of a subset (which can be empty) excluding at least one element. Valid outputs encompass any numbers from $\{0, 1, \dots, k - 1\}$ that are absent from the enumeration.

The investigation into choice principles extends to finite sets and convex sets in [134], where Le Roux and Pauly establish the dimension of convex sets that can be characterised by the cardinality of finite sets. Le Roux and Pauly delve into finding zeros of functions with finite local extrema. They employ an algorithm to compute a fixed finite number of real numbers that will include all zeros of the function at hand. Their work reveals that finding a zero of a continuous function with finite local minima is Weihrauch reducible to $C_{\{1, \dots, 3^n\}}$. In addition, they establish that $C_{\{1, \dots, n\}}$ is Weihrauch reducible to finding roots of polynomials of degree $2n$.

The choice problem has been subject to extensive study through numerous variants, often involving restrictions to closed subsets possessing specific additional properties.

- Definition 2.21** (Variants of Choice)
1. Unique choice: $UC_{\mathbf{X}}$ is $C_{\mathbf{X}}$ restricted to singletons,
 2. Connected choice: $CC_{\mathbf{X}}$ is $C_{\mathbf{X}}$ restricted to connected sets,
 3. Pathwise connected choice: $PWCC_{\mathbf{X}}$ is $C_{\mathbf{X}}$ restricted to pathwise connected sets,
 4. Convex choice: $XC_{\mathbf{X}}$ is $C_{\mathbf{X}}$ restricted to convex sets,
 5. Positive choice: $PC_{\mathbf{X}}$ is $C_{\mathbf{X}}$ restricted to sets with positive measure,
 6. All-or-unique choice: $AoUC_{\mathbf{X}}$ is $C_{\mathbf{X}}$ restricted to sets of the form $\{x\}$ or \mathbf{X} ,
 7. All-or-co-unique choice: $ACC_{\mathbf{X}}$ is $C_{\mathbf{X}}$ restricted to sets of the form $X \setminus \{x\}$ or \mathbf{X} ,
 8. Co-finite choice: $CFC_{\mathbf{X}}$ is $C_{\mathbf{X}}$ restricted to co-finite sets.

Drawing special attention to the AOUC principle, which holds immense significance in this research, $AoUC_{[0,1]}$ operates with an abstract input that delineates valid solutions $x \in \{0, 1\}^{\mathbb{N}}$ as follows: Initially, the entire set $\{0, 1\}^{\mathbb{N}}$ is considered a valid solution. This state can persist indefinitely (termed the ‘all’ case), or at some juncture, we may receive information that there exists a sole correct answer, accompanied by knowledge of what that answer is (referred to as the ‘unique’ case).

It’s worth noting that another choice problem, denoted as $K_X : \subseteq \mathcal{K} _ (X) \rightrightarrows X, K \vdash K$ and is referred to as ‘compact choice’ exists. Unlike the other choice problems, this one not only constrains C_X to compact sets but also augments the input information. Specifically, the input set K is described as a compact set, defined in previous sections.

Any single-valued function f that can be computed from compact choice and another function g can already be computed by g alone [20]. This provides the possibility to ‘divide’ by compact choice in some situations. The notion of mind changes is introduced, wherein a function is computable with finitely many mind changes if it can be computed

on a Turing machine that revises its output at most finitely many times for each particular input.

We present an overview of significant classes of functions that can be delineated through suitable versions of the choice principle.

Notions of Computability: Let f be a problem. All the given properties of f are invariant.

1. $f \leq_W C_{\mathbb{N}} \iff f$ is computable with finitely many mind changes.
2. $f \leq_W C_{2^{\mathbb{N}}} \iff f$ is non-deterministically computable.
3. $f \leq_W PC_{2^{\mathbb{N}}} \iff f$ is Las Vegas Computable.

Numerical quantities that remain invariant under Weihrauch reducibility can also provide valuable insights. In this regard, another useful approach involves considering the number of mind changes necessary to compute a problem. Denoted as $\text{mind}(f)$, this value represents the minimum number $n \in \mathbb{N}$ that a Turing machine with two-way output requires to compute f , ensuring a maximum of n mind changes across all inputs (if such a number exists).

Furthermore, our exploration delves into computation with a finite number of mind changes. Our journey begins with a model of computation where the machine progressively generates more and more digits of the infinite code for the desired output. However, we augment this process by granting the machine the capability to erase all previously written digits and commence anew. To maintain the integrity of the output, this erasure ability can be invoked only finitely many times. The result is an upper bound that sheds light on the number of mind changes needed for the computation. A problem is solvable with many mind changes if and only if it is Weihrauch reducible to C_n .

A Las Vegas machine employs random coin flips to aid its computation. While it can halt at any point, reporting a fault and aborting, if it continues indefinitely, it must produce a valid output. For each input, the probability (based on the coin flips) of generating a correct output needs to be positive (without necessitating a global positive lower bound). Therefore, a problem is considered Las Vegas computable if it can be Weihrauch-reduced to WWKL.

By relaxing the requirement that an incorrect output must be reported during computation, we encounter Monte Carlo machines. These machines also employ random coin tosses and adhere to the condition that any completed output must be accurate. However, they can fail by simply ceasing to produce output. A problem is classified as Monte Carlo computable if it can be Weihrauch-reduced to WWKL', which is prime with C'_n .

2.5 Reverse Mathematics Principles

Reverse mathematics seeks to answer the question of which axioms are appropriate for mathematics. An extensive introduction to this topic is provided by Simpson [143]. If

a mathematical theorem is proved from an appropriately weak set existence axioms, then the axioms will be logically equivalent to the theorem. This book goes into depth on many of these axioms such as RCA_0 , WKL_0 , ACA_0 , $\text{ATR}-0$ and Π_1^0 . Several key principles and concepts play significant roles in understanding the logical strengths and interrelationships of various mathematical theorems.

A newly published textbook [51] delves into the intricacies of demonstrating mathematical theorems and resolving mathematical conundrums. The book adopts a contemporary approach to reverse mathematics by interlacing reductions rooted in computability theory with proofs grounded in formal arithmetic. This fusion allows for an assessment of the intricacy of theorems and problems across the entire spectrum of mathematics. The work serves as a comprehensive primer on reverse mathematics, presenting a thorough exploration of the reverse mathematics pertaining to combinatorics. Furthermore, it encapsulates pivotal findings and approaches from the most recent twenty-year period. Each chapter is enriched with a plethora of exercises spanning various levels of difficulty, thus complementing the content effectively.

Recursive Comprehension Axiom, often denoted as RCA_0 , serves as the foundational system in the field of reverse mathematics. It is a subset of second-order arithmetic designed to align with computable and constructive mathematics principles. In RCA_0 , the comprehensive schema is modified to apply exclusively to Δ_1^0 sets. Basically, $\exists X \forall n (n \in X \leftrightarrow \phi(n))$. However, this must be expressible as a Δ_1^0 formula. Additionally, the induction principle is restricted to Σ_1^0 formulas.

Arithmetical comprehension (ACA_0) is a formal system defined similarly to RCA_0 , although it is, in fact, much stronger. It incorporates all the logical and arithmetic axioms of standard Peano arithmetic, in addition to including a comprehensive schema covering all arithmetic formulas. ACA_0 enables us to construct the set of natural numbers satisfying arbitrary arithmetical formulas (with no bounded set variables).

Hirschfeldt also contributes to the exploration of reverse mathematics and computable mathematics in his work [74]. He delves into a range of phenomena and techniques. The book includes an introduction to König's Lemma and computability. They focus on Ramsey's Theorem, with particular attention on stability, cohesiveness and Mathias forcing. The Weak König's Lemma (WKL) states that every infinite and definable tree admits an infinite path, forming a foundational principle that has implications across various mathematical areas. WWKL, an abbreviation for Weak Weak König's Lemma, asserts the presence of an infinite path within an infinite binary tree, demonstrating a weaker version of the classical König's Lemma.

Arithmetical transfinite recursion (ATR_0) is a subsystem of second-order arithmetic that is logically stronger than ACA_0 . This system can be described as an extension of ACA_0 by allowing the transfinite iteration of the Turing jump operator along any countable well-ordering. ATR_0 is considered impredicative, and has the proof-theoretic ordinal Γ_0 , the supremum of that of predicative systems.

In the realm of reverse mathematics, both Π_1^0 and Π_1^1 classes assume pivotal roles, contributing to the classification and comparison of mathematical principles and theorems based on their logical properties. Π_1^0 classes represent sets of natural numbers that can be defined by specifying conditions universally applicable to all natural numbers. On

the contrary, Π_1^1 classes extend the hierarchy of descriptive set theory and introduce a higher degree of complexity. A set belongs to a Π_1^1 class if its characterisation entails conditions that universally apply to all natural numbers, followed by conditions that existentially hold for certain additional parameters. This incorporation of both universal and existential quantification empowers Π_1^1 classes to encapsulate more intricate mathematical properties and relationships in contrast to the capabilities of Π_1^0 classes.

One of the strongest subsystems is $\Pi_1^1 - \text{CA}_0$, which can be defined as the subsystem consisting of RCA_0 together with the comprehension scheme for Π_1^1 formulas. $\Pi_1^1 - \text{CA}_0$ proves the induction scheme for Π_1^1 formulas and for Σ_1^1 formulas as well.

The connection between reverse mathematics and computable analysis has been a long-standing investigation. A recent development in this field focuses on first-order strength, with the most robust outcome referred to as the first-order part of the theorem. In their work [58], the concept of the first-order part of a problem is introduced, capturing the strongest ‘number-theoretic’ problem that is Weihrauch reducible to the given problem. Dzhafarov, Solomon and Yokoyama demonstrate that if a theorem is undiagonalisable (such as COH, FIP, and Π_1^0G), its first-order part as a problem becomes trivial. This insight allows them to conclude that these theorems are uniformly computably true.

2.6 Translating Between Spaces of Subsets

It is important to note this is new research ([48]). Section 2.2 introduces various subsets such as closed, compact and overt. This raises the question of when we can translate between two of these subsets. We know that Hausdorff compact subsets are also closed and that overt subsets have some of the same elements as closed subsets. Therefore, how can we use this to find the Weihrauch computability of these translations. According to the literature, closed to compact subsets has not been studied yet. In this work, we answer this open question.

A name for some set $A \in \mathcal{K}(\mathbb{R}^n)$ can be thought of as a name for $A \in \mathcal{A}(\mathbb{R}^n)$ along with some $M \in \mathbb{N}$ such that $A \subseteq [-M, M]^n$ (e.g. [164]). Consequently, the translation $\text{id} : \subseteq \mathcal{A}(\mathbb{R}^n) \rightarrow \mathcal{K}(\mathbb{R}^n)$ merely involves determining an appropriate bound M . To classify its Weihrauch degree, we compare it to the following: The principle $\Pi_1^0 \text{Bound}$ takes a finite closed subset of \mathbb{N} as input and produces an upper bound for it. Its counterpart $\Pi_1^1 \text{Bound}$ was introduced and explored in [65]. Using $\Pi_2^0 \text{C}_{\mathbb{N}}$ we denote the choice operator for Π_2^0 -subsets of \mathbb{N} . This entails being given a Π_2^0 -code for some non-empty $A \subseteq \mathbb{N}$ and needing to provide an $n \in A$. Then $\Pi_2^0 \text{C}_{\mathbb{N}}|_{\text{upw-cl}}$ represents the restriction of $\Pi_2^0 \text{C}_{\mathbb{N}}$ to upwards-closed sets, i.e. those $A \subseteq \mathbb{N}$ where $n \in A$ and $n < m$ implies $m \in A$.

Theorem 2.22 [jww Arno Pauly] The following are Weihrauch equivalent:

1. $\text{id} : \subseteq \mathcal{A}(\mathbb{R}^n) \rightarrow \mathcal{K}(\mathbb{R}^n)$
2. $\text{id} : \subseteq \mathcal{A}(\mathbb{N}) \rightarrow \mathcal{K}(\mathbb{N})$
3. $\Pi_1^0 \text{Bound} : \subseteq \mathcal{A}(\mathbb{N}) \rightrightarrows \mathbb{N}$

4. $\Pi_2^0 C_{\mathbb{N}}|_{\text{upw-cl}} : \subseteq \Pi_2^0(\mathbb{N}) \rightrightarrows \mathbb{N}$

Proof. 1. $(\text{id} : \subseteq \mathcal{A}(\mathbb{R}^n) \rightarrow \mathcal{K}(\mathbb{R}^n)) \leq_W \Pi_1^0 \text{ Bound}$

Given an element $A \in \mathcal{A}(\mathbb{R}^n)$, we have the capability to compute the set $\{k \in \mathbb{N} \mid A \cap ([-k-1, k+1]^n \setminus (-k, k)^n) \neq \emptyset\} \in \mathcal{A}(\mathbb{N})$. This is feasible because each $[-k-1, k+1]^n \setminus (-k, k)^n$ is at our disposal as a compact set. When A is bounded, the resulting set is finite. Any upper bound for this set (provided by $\Pi_1^0 \text{ Bound}$) serves to complete the $\mathcal{A}(\mathbb{R}^n)$ -name for A to a $\mathcal{K}(\mathbb{R}^n)$ -name for A .

2. $(\text{id} : \subseteq \mathcal{A}(\mathbb{N}) \rightarrow \mathcal{K}(\mathbb{N})) \leq_W (\text{id} : \subseteq \mathcal{A}(\mathbb{R}^n) \rightarrow \mathcal{K}(\mathbb{R}^n))$

We can embed $\mathcal{A}(\mathbb{N})$ into $\mathcal{A}(\mathbb{R}^n)$ by mapping A to $\{(k, \dots, k) \in \mathbb{R}^n \mid k \in A\}$.

3. $(\text{id} : \subseteq \mathcal{A}(\mathbb{N}) \rightarrow \mathcal{K}(\mathbb{N})) \equiv_W \Pi_1^0 \text{ Bound}$

By its definition, $\Pi_1^0 \text{ Bound}$ is the very task of finding the upper bound that differentiates a $\mathcal{K}(\mathbb{N})$ -name of a finite subset of \mathbb{N} from a $\mathcal{A}(\mathbb{N})$ -name for the same set.

4. $\Pi_1^0 \text{ Bound} \leq_W \Pi_2^0 C_{\mathbb{N}}|_{\text{upw-cl}}$

For $M \in \mathbb{N}$ to be an upper bound for $A \in \mathcal{A}(\mathbb{N})$ is stating that for all $t > M$, $t \notin A$. This is a Π_2^0 -statement, and the set of solutions is clearly upwards closed.

5. $\Pi_2^0 C_{\mathbb{N}}|_{\text{upw-cl}} \leq_W \Pi_1^0 \text{ Bound}$

A set $A \in \Pi_2^0(\mathbb{N})$ can be represented by a sequence $(p_{n,i}) \in \mathbf{2}^{\mathbb{N} \times \mathbb{N}}$ such that $A = \{n \in \mathbb{N} \mid \exists^\infty i \in \mathbb{N} p_{n,i} = 1\}$. From this representation, we can derive the closed set $B = \{(n, i) \in \mathbb{N}^2 \mid p_{n,i} = 1 \wedge \forall j > i p_{n,i} = 0\}$. If A is upwards-closed, B is finite. Additionally, under the assumption of a reasonably defined pairing function, any strict upper bound for B also serves as a strict upper bound for the complement of A , hence belonging to A . Consequently, by applying $\Pi_1^0 \text{ Bound}$ to B , we acquire a value $n \in \mathbb{N}$, implying that $n + 1 \in A$. □

Definition 2.23 ([65]) Let \mathbf{X} be a represented space and $f : \subseteq \mathbb{Y} \rightrightarrows \mathbb{Z}$ be a multivalued function. We define $\text{Det}_{\mathbf{X}}(f) : \subseteq \mathbb{N}^{\mathbb{N}} \times \mathbb{Y} \rightarrow \mathbb{X}$ by

$$\text{Det}_{\mathbf{X}}(f)(p, y) = x \iff (\forall s \in \delta_{\mathbb{Z}}^{-1}(f(y)))(\delta_{\mathbf{X}}(\Phi_p(s)) = x),$$

where $\Phi_{(\cdot)}$ is an universal Turing functional. The domain of $\text{Det}_{\mathbf{X}}(f)$ is maximal for this to be well-defined. We write $\text{Det}(f)$ for $\text{Det}_{\mathbb{N}^{\mathbb{N}}}(f)$.

$\text{Det}(f)$ is the greatest Weihrauch degree of a single-valued function with codomain $\mathbb{N}^{\mathbb{N}}$ which is Weihrauch reducible to f . We can conclude that:

Corollary 2.24 $\text{Det}(\text{id} : \subseteq \mathcal{A}(\mathbb{R}^n) \rightarrow \mathcal{K}(\mathbb{R}^n)) \equiv_W C_{\mathbb{N}}$.

Proof. This follows from [65, Lemma 4.13] and Theorem 2.22. For the reverse direction, we recall that $\text{UC}_{\mathbb{N}} \equiv_W C_{\mathbb{N}}$, and that given some $\{n\} \in \mathcal{A}(\mathbb{N})$, some upper bound $M \geq n$ suffices to compute n . □

Proposition 2.25. $\text{isInfinite}_{\mathbb{S}} \leq_{\text{W}} \Pi_1^0 \text{Bound}$

Proof. Given some $p \in \{0, 1\}^{\mathbb{N}}$, we can compute the set of positions in p where a 1 appears for the last time as an element of $\mathcal{A}(\mathbb{N})$ - it is either a singleton or the empty set. If we obtain a bound N for this set from $\Pi_1^0 \text{Bound}$, then we know that we will encounter a 1 in p after position N iff p contains infinitely many 1s. \square

Proposition 2.26. $\Pi_1^0 \text{Bound} \times \Pi_1^0 \text{Bound} \leq_{\text{W}} \Pi_1^0 \text{Bound}$

Proof. We can just take the union of the two closed sets we obtain as an input on the left, and use it as an input for the single copy of $\Pi_1^0 \text{Bound}$. Any resulting upper bound is a correct output for both instances. \square

Chapter 3

Game Theory and Hypergame Theory

Contents

3.1	Game Theory	27
3.2	Machine Learning and Game Theory	31
3.3	Game Theory and Complexity	33
3.4	Hypergame Theory	35

3.1 Game Theory

Game theory is a branch of mathematics that studies how individuals and entities make decisions in strategic situations where their choices affect each other's outcomes. It involves analysing the interactions between players, their choices, and the outcomes they collectively achieve, often with the goal of understanding optimal decision-making strategies and predicting the likely results of various scenarios. Standard sources on game theory include [104] and [117]. My Masters Thesis also explored Game theory and games with incomplete information [46]. In the context of cybersecurity (needed for Chapter 8), the dilemma can be conceptualised as a competition between attackers and defenders vying for control over a system.

There are two primary representations in Game theory; extensive-form (tree form) and normal-form (matrix form). Extensive-form presents the information as a decision tree, where actions are selected from the root to a leaf, culminating in a payoff for the entire path. In our paper, we focus on normal-form games, where players' strategies are depicted as the matrix's rows and columns. The payoffs are then located at the intersection of these strategies.

Definition 3.1 A game in *normal-form* is an ordered triple, $G = (N, (S_i)_{i \in N}, (u_i)_{i \in N})$ in which:

- $N = \{1, 2, \dots, n\}$ is a finite set of players.
- S_i is the set of strategies of player i , for every $i \in N$.
Denote the set of all vectors of strategies by $S = S_1 \times S_2 \times \dots \times S_n$.
- $u_i : S \rightarrow \mathbb{R}$ is a utility function.

The utility function u_i associates each vector of strategies, $S = (S_i)_{i \in N}$ with the payoff $u_i(S)$ to player i .

In the realm of game theory, the participants are referred to as agents or players. Players are defined as a set of rational agents who take actions based on a set of available choices, which means they will always act to their own advantage. The set of players is defined as $n \in N$. In order to define a game, the rules need to be defined. These are the moves each player can make, typically referred to as the strategies of each player.

An example of a normal-form game is shown in Table 3.1. In this scenario, the defender (d) is tasked with protecting two targets against the actions of the attacker (a). The game entails the following choices: the defender can opt to defend target 1 (d_1) or target 2 (d_2), while the attacker can decide to attack target 1 (a_1) or target 2 (a_2). The payoffs within the cells represent the respective payoffs for the defender and attacker, where a higher value indicates a more favorable outcome.

	a_1	a_2
d_1	1,0	-10,10
d_2	-2,2	1,0

Table 3.1: A basic 2-player game in normal-form, with an attacker (a) and defender (d)

The measure of satisfaction of the player is defined as the utility of a player or the payoff. This utility represents the reward obtained by selecting a specific course of action, which can signify projected financial losses or damages. The rational decision-making of the players is driven by utility as it implies a ranking or priority among the options. Rooted in decision theory, the utility function allows the player to calculate the expected utility based on their preferences when set against an indifferent player. Utility, u_{ij} , is the given payoff for player i for taking action j , where $u_{ij} \in U$, U is the set of all utility functions.

Considering the game in Table 3.1, there is a clear preference for the defender to play d_2 , as the worst outcome would only be -2 instead of the -10 of d_1 . The ultimate payoff in a game remains undisclosed until both players have executed their actions, as a player's reward hinges on their action in conjunction with the actions of all other participants.

In game theory, players can adopt either pure strategies or mixed strategies in terms of the process by which the player chooses between sets of actions. Pure strategies are the complete action taken by a player, they can be a single choice or action in a simple game or a set of decisions. Importantly, pure strategies are uninterruptable, once all players have chosen a strategy, all actions are executed simultaneously.

Definition 3.2 A pure strategy for a player i is given as s_i and the set of possible strategies is given by $S_i = \{s_i^1, \dots, s_i^m\}$ where s_i^m is the m^{th} strategy for player i .

For the game set out in Table 3.1, each player has two strategies target 1 and 2, the notation for this would give $S_d = \{s_d^1, s_d^2\}$ and $S_a = \{s_a^1, s_a^2\}$.

On the other hand, mixed strategies are a probability distribution across various pure strategies that a player views as viable choices.

Definition 3.3 A mixed strategy, σ , for player i is given as $\sigma_i = (p_i^1, \dots, p_i^m)$ where p_i^m represents the probability of player i playing pure strategies m .

Each mixed strategy is subject to the constraint $\sum_j p_i^j = 1$. The utility for a mixed strategy is given as $u_i(\sigma_i, \sigma_{-i})$.

The utility (u_i) of a player is the payoff assigned to that player based on the actions of all participants.

Definition 3.4 A utility is defined as $u_i(s_i, s_{-i})$, where s_i represents the strategy chosen by player i and s_{-i} signifies the strategy of all other players.

A utility function allows us to make judgements and analyse preferences of strategies. To achieve this, a *preference ordering* is needed. Ordered preferences show how different outcomes satisfy the player's preferences. The assumption of transitivity is an example of this, if a player prefers outcome x to y and prefers y to z it is consistent if they prefer x to z .

Definition 3.5 Let O be a set of outcomes and \succsim be a complete, reflexive, transitive preference relation over O . A function $u : O \rightarrow \mathbb{R}$ is called a *utility function* representing \succsim if for all $x, y \in O$ $x \succsim y \Leftrightarrow u(x) \geq u(y)$.

Definition 3.5 implies that u is a function associating an outcome with a real number, where the higher the real number associated with it, the more preferred the relative outcome. The utility numbers are known as *ordinal utility* as they convey nothing more than information on the ordering of preferences [70].

In the game from Table 3.1 if the defender opts for d_2 the payoff will be based on the action of the attacker. If the attacker plays a_1 the defender's utility would be $u_d(d_2, a_1) = -2$.

Games can be categorised based on the information players possess regarding the actions of others. Perfect information implies that all players are fully aware of every move that has transpired. On the other hand, imperfect information is when players lack knowledge of each other's previous moves. Additionally, games can be classified according to the extent of information about the game's structure and participants' utility functions. Complete information denotes the structure of the game and the payoffs/utility functions of the players are common knowledge. An example of this is chess as all of the rules are laid out fully before the game begins. Conversely, incomplete information is when players lack full knowledge of the game's structure or other players' utility functions or strategies.

The optimal solution within a game is defined as the strategy from which no player can achieve a higher utility by deviating. This optimal strategy for all players is referred to as an equilibrium. At this equilibrium point, no player is incentivised to alter their strategy, denoted as σ^* . This is given by $u_i(\sigma_i^*, \sigma_{-i}) \geq u_i(\sigma'_i, \sigma_{-i})$ where σ'_i is any alternative strategy to σ_i and σ_{-i} is the best response strategy amongst all other players.

Definition 3.6 A strategy vector $s^* = (s_1^*, \dots, s_n^*)$ is a *Nash Equilibrium* if for each player $i \in N$ and each strategy $s_i \in S_i$, the following is being satisfied:

$$u_i(s^*) \geq u_i(s_i, S_i^*).$$

The payoff vector $u(s^*)$ is the *equilibrium payoff* corresponding to the Nash equilibrium s^* .

The expected payoff for a mixed strategy is defined as the sum of each utility given the probability that the utility is achieved. The expected utilities for the attacker and defender can be shown by $u_a = \sum_j p_a^j u_a(\sigma_a, \sigma_d)$ and $u_d = \sum_j p_d^j u_d(\sigma_d, \sigma_a)$.

Definition 3.7 A *Nash Equilibrium* for a two-player game is defined as

$$\max_{\sigma_a} \left(\sum_j p_a^j u_a(\sigma_a, \sigma_d) \right) \text{ and } \max_{\sigma_d} \left(\sum_j p_d^j u_d(\sigma_d, \sigma_a) \right).$$

Given the game in Table 3.1 and the optimal strategies $\sigma_d^* = (0.333, 0.667)$, meaning the defender plays d_1 with $p_1 = 0.333$ and $p_2 = 0.667$, meanwhile the attacker's optimal strategy is $\sigma_a^* = (0.667, 0.333)$. In order to calculate the expected payoff for d_1 , calculate the individual utilities:

$$\begin{aligned} u_d(d_1, a_1) p_d(d_1) p_a(a_1) &= 1 \times 0.333 \times 0.667 = 0.222 \\ u_d(d_1, a_2) p_d(d_1) p_a(a_2) &= -10 \times 0.333 \times 0.333 = -1.110 \\ u_d(d_2, a_1) p_d(d_2) p_a(a_1) &= -2 \times 0.667 \times 0.667 = -0.890 \\ u_d(d_2, a_2) p_d(d_2) p_a(a_2) &= 1 \times 0.667 \times 0.333 = 0.222 \end{aligned}$$

Then sum the equations for the final result of the expected utility of d_1 being -1.556.

Game theory involves the study of interactions among rational agents aiming to maximise their gains. It uses diverse methods to model these interactions, capturing complex scenarios. In the case of cyber-related responsibilities, instead of replacing decision-making processes, game-theoretic approaches should compliment and enhance them.

At the core of game theory is the concept of 'regret minimisation.' This strategy seeks to minimise the gap between the outcomes achieved through a player's chosen actions and those achievable by different strategies. Regret minimisation strives to refine decision-making strategies by learning from past actions and their outcomes. A paper exploring regret minimisation in games with incomplete information can be found in [170].

Within this thesis, several other games are referenced to provide context and illustrate key concepts. One of these is the Gale-Stewart games, which serve to highlight the applications of Π_1^0 -classes. In a Gale-Stewart game, two players take turns selecting elements a_i from the set $\{0, 1\}$ [145].

Another notable game discussed is the Infinite Repeated Prisoner's Dilemma. This scenario involves an infinitely repeated situation where decision-makers continually face the prisoner's dilemma. In this dilemma, each individual is incentivised to choose in a manner that results in a suboptimal outcome for the collective group. Many approaches have been investigated as to the strategies of the game such as Tit-For-Tat and Grim trigger [50].

3.2 Machine Learning and Game Theory

3.2.1 Game Theory and Learning

3.2.1.1 Adversarial Learning

Machine learning is a methodology designed to extract meaningful patterns from large amounts of data. Adversarial learning delves into the connection between a learning system and the adversary, typically characterised by opposed preferences. In their survey, Zhou, Kantarcioglu and Xi explore game theoretic models used in adversarial learning [169]. This area involves two main types of games: simultaneous games and sequential games.

The simultaneous games are commonly represented as zero-sum games, where the learner searches for an optimal strategy that maximises their expected payoff while anticipating the adversary to do the opposite. On the other hand, sequential games offer the learner opportunities to capitalise on first-mover advantages by controlling the cost the adversary incurs to evade detection.

To model the roles of both the adversary and the learner, Stackelberg games are employed. In this framework, the adversary takes on the role of the leader, playing the most rational strategy at each move. Meanwhile, the learner acts as the leader and simulates the antivirus software. The adversary is more likely to probe for the software classifiers.

Bayesian Stackelberg games and Single-Leader-Multifollower games are also considered, as there may be different adversaries that need to be taken into account, or many adversaries at the same time.

3.2.1.2 Multi-agent Reinforcement Learning

Reinforcement learning draws inspiration from animal learning, where good behaviours are rewarded and unfavourable behaviours are punished. Unlike supervised learning, which involves a teacher, reinforcement learning revolves around the learner making decisions and evaluating the outcomes. In their study, Tuyls and Nowé explored various learning techniques and compared them [156]. Their investigation revealed that while independent learners' performance deteriorates with increased information, they excel in

revelation games. However, they are not guaranteed to converge to the Pareto Optimal Nash equilibrium. In contrast, revelation learners could not overcome penalties in the penalty game. Frequency Maximum Q technique (FMQ) learners achieved a convergence rate of 100% in identical payoff games, though assurance was lacking in other game scenarios. However, when games have an optimal group utility that differs from the agents' personal utility, FMQ learners fail to reach the optimal solution. Exploring Selfish Reinforcement Learning (ESRL) learners, while reaching the Pareto optimal solution in all games, generally required more time to converge compared to FMQ.

Evolutionary game theory's insight into multi-agent learning was highlighted by Tuyls and Parsons [157]. This theoretical framework offers a means to analyse the behaviour of multi-agent systems, particularly in scenarios where agents lack complete information about each other's goals and the game state. Given the dynamic nature of multi-agent systems, where agents can alter their behaviours over time, evolutionary game theory provides a robust foundation for understanding these iterative dynamics. Tuyls and Parsons findings indicate that the normative agenda has limited contributions to the goals of multi-agent learning, while the descriptive agenda holds the potential for building systems involving real agents and human interactions.

3.2.1.3 Boosting Learning Algorithms

Boosting stands as a general method for improving the accuracy of any given learning algorithm, an explanation of this and its connection to game theory was completed by Schapire [140]. The AdaBoost algorithm maintains a distribution or set of weights over the training set. Initially, all the weights are equal but in each round, the weights of incorrectly classified examples are increased. This compels the learner to focus on the hard examples.

Classical game theory involves two-player zero-sum games which can be presented as a game matrix. Boosting can be viewed as the repeated play of a particular game matrix. The boosting algorithm takes on the role of the row player, while the base learner assumes the position of the column player. A well-known example of boosting and game theory is von Neumann's famous minmax theorem. This theorem explains that the AdaBoost theorem, at least, has the potential for success.

3.2.2 Game Theory and Artificial Intelligence

Both game theory and Artificial Intelligence (AI) revolve around 'intelligent' agents navigating a complex world. Despite their profound connections, the research directions of the two disciplines have often diverged. Tennenholtz looks into this intriguing intersection [153]. The discussion encompasses topics such as reasoning within distributed systems, incorporating communication and rationality constraints, reinforcement learning, modelling agents as expected utility maximisers and Savage axiomatisation.

Llewyn ventured into the realm of AI by engaging with the classic game rock-paper-scissors, employing Q-Tables and Markov Chains [102]. The Q-learning approach aims to assign values to actions associated with specific states by observing their outcomes. The

resulting Q-table features rows representing states and columns representing available actions for each state, yielding a 3x3 table. To enhance the agents' decision-making abilities, a 'perception' element was introduced, allowing them to consider past moves. This adaption of adding one move backwards leads to a 9x3 table. Meanwhile, Markov chains, representing events occurring over time, were employed. The study utilised a finite state space Markov chain, with each starting state initialised at a 33.3% probability.

The implementation of the game was completed by coding a game environment where each agent had their own modules, enabling the repetition of games multiple times. It was found that the agents' ability to change what information is held about their opponents was the most important aspect for securing victories. Intriguingly, while the capacity to consider a larger history of past games seemed advantageous, it paradoxically slowed down agents' adaptation to shifts in opponents' strategies.

Attala also explored this game, employing both Markov Chains and Support Vector Machines (SVM) [6]. SVMs identify the optimal separating Hyperplane that distinguishes classes in the training data. They utilised n moves from the opponent as training data, recording both the moves and their outcomes. These were implemented using Python, where each outcome was represented by 0,1,2. This allows actions to be compared using $+1 \pmod{3}$ for wins and $-1 \pmod{3}$ for losses. Notably, SVMs with non-linear kernels demonstrated heightened effectiveness, adept at accommodating both anomalous and linear behaviour. Their Python implementation highlighted that first-order Markov chains, basing decisions solely on the previous game, experienced exponentially reduced prediction accuracy as block length exceeded 1. Exploring Nash equilibria in SVM games, C-values generally converged to 1, with occasional occurrences of 0.7, 0.9, and 1 at equal probabilities.

Transforming an extensive-form game into a Partially Observable Markov Decision Process (POMDP) model for a single player is feasible under the assumption of a fixed opponent [115]. POMDPs address scenarios rife with state uncertainty, providing observations that offer hints about the true state. Oliehoek showed how to convert an 8-card poker game into a POMDP under the assumption that the opponent is fixed and known. The solution process of the POMDP entails generating all possible beliefs and their transition probabilities, effectively constructing a Markov Decision Process (MDP). This MDP is solved using exact value iteration.

3.3 Game Theory and Complexity

3.3.1 Polynomial Parity Argument on Directed Graphs

An interesting Total Function in NP is called Polynomial Parity Argument (PPA). PPA is given a finite graph consisting of lines and cycles, there is an even number of endpoints. The class Polynomial Parity Argument Directed (PPAD) is defined using directed graphs based on PPA. It involves solving a problem on an exponential-sized directed graph, with each node having in-degree and out-degree at most one. This is described through a polynomial-time computable function $f(v)$ that outputs the predecessor and successor

of v , and a vertex s with a successor but no predecessors, and finds a $t \neq s$ that either has no successors or predecessors.

For an introduction to the complexity class PPAD and the complexity of finding Nash equilibria, refer to [118]. In this work, Papadimitriou explains the rationale behind not treating the Nash equilibria problem as an NP-complete problem. Unlike NP-complete problems, every game is guaranteed to have a Nash equilibrium whereas NP-complete problems may or may not exist. After introducing succinct and graphical representations of games, the paper proceeds to present a proof that finding a mixed Nash equilibrium is PPAD-complete.

The problem of finding a Nash equilibrium in a two-player game with rational payoffs falls under the complexity class PPAD [40] where it is established to be PPAD complete. The PPAD is a subset of the class Total Function Problems (TFNP). This can be extended to show that finding an approximate equilibrium in a non-cooperative game among three or more players is also in PPAD.

Goldberg and Papadimitriou contribute to the understanding of the complexity of Nash equilibrium by revealing that it doesn't form an infinite hierarchy; rather, it collapses to the fourth level [66]. In their paper, they outline how to reduce graphical games to strategic form games and vice versa. Therefore, by applying both reductions to a strategic form or graphical game, they obtain a game of the same type that is at least as hard to solve, despite having restrictions on its structure.

The search problem for an r -player game in strategic form with a binary integer is defined as r -Nash within [54], where they demonstrated that r -Nash falls within PPAD. A search problem is considered PPAD-complete if all problems in PPAD reduce to it. Daskalakis, Goldberg and Papadimitriou show that 4-Nash is PPAD-complete in their main theorem of the paper, (these results use sets of mixed strategies which are ϵ -Nash equilibrium).

The three aforementioned papers establish that the general Nash equilibria problem for both strategic form games and graphical games can be reduced to two-player games. Daskalakis, Fabrikant and Papadimitriou extend these results to all known representations of games and to more sophisticated concepts of equilibrium [53]. They achieve this using succinct games and expected utility, where the latter involves computing the expected utility of a player given a mixed strategy profile.

Papadimitriou and Roughgarden provide a constructive proof in [119] that every game possesses a correlated equilibrium. They also introduced the notion of the polynomial expectation property. Their work demonstrated that optimal correlated equilibria of anonymous games can be computed in polynomial time, along with graphical games with bounded tree width. A corollary shows a contrast to symmetric games with a constant number of players by combining various results to show that it is PPAD-complete to compute a symmetric Nash equilibrium.

3.3.2 Fixed-Point Complexity

FIXP, denoting the class of fixed point problems, involves expressing search problems as fixed point problems for functions represented by polynomial-sized algebraic circuits

using operations like $\{+, -, *, /, \max, \min\}$ along with rational constants.

The Fixed-Point complexity class (FIXP) is a class of real-valued total search problems introduced within [59]. They show that Nash exactly characterises the search problems that can be cast as fixed points of functions represented by algebraic circuits. The computation of Nash equilibria for three or more players is complete for FIXP.

3.4 Hypergame Theory

Hypergames are a system comprised of a set of games with incomplete information. Each game in the system shows the game from one player's viewpoint. Bennett [16] introduces a hypergame.

Definition 3.8 (Bennett's Definition) An n -person hypergame in normal form is a system consisting of:

- a set P_n of n elements (the players of the hypergame),
- for each $p, q \in P_n$ a non-empty finite set S_p^q (the set of strategies available for p),
- for each $p, q \in P_n$ an ordering relationship O_p^q defined over the product space $S_1^q \times \dots \times S_n^q$, (p 's preference ordering, perceived by q).

The sets S_1^q, \dots, S_n^q are player q 's strategy matrix. The set P_n and the orderings P_1^q, \dots, O_n^q are player q 's game within the hypergame, denoted G^q . The hypergame H , is considered the set of n games G^1, \dots, G^n .

Hypergames are valuable for resolving situations characterised by miscommunications about events or available strategies. Bennett further delved into this concept by examining various case studies of hypergames [15]. He also provided a schematic illustration of strategy mappings for a simple two-player hypergame. This includes the links between the two perceptual strategy spaces, e.g. the links between the two player versions of the game. Bennett goes over the straightforward hypergame example called 'The Fall of France'.

Example 3.9 (The Fall of France) In 'Fall of France' there are two players, the Germans and the Allies, resulting in two games within the hypergame (seen in Figure 3.1). The Allies' game encompasses the evident strategies that both players could pursue. Conversely, the Germans introduce a 'surprise' strategy which the Allies are unaware of.

The values in the cells represent the preferences of the Germans and the Allies in that order, with higher values indicating stronger preferences. The Nash equilibrium in the Allies' game is (AN,MN), while in the German game is (AN,N+C). Despite the similarity in outcomes between these distinct games, this analysis doesn't consider the scenario where the Germans are aware of the Allies' lack of knowledge about the extra strategy. In such a situation, the Germans can take this perspective into account, introducing 'higher-order' beliefs. These beliefs enable the Germans to anticipate that the Allies would most likely take the 'Move North' strategy. Consequently, the Germans

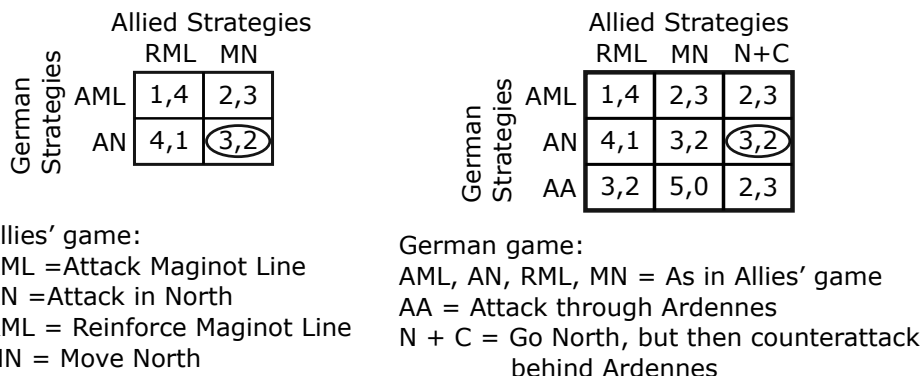


Figure 3.1: A simple two-player hypergame, ‘The Fall of France’

can deviate from the Nash equilibrium’s ‘Attack in North’ and instead choose ‘Attack through Ardennes’, securing a robust victory (which indeed happened).

In addition to illustrating the connections between the perceptual space and the ‘basic’ space, which is the ‘standard’ normal-form two-player game, Bennett also emphasises the need for a systematic framework for hypergames to prevent the occurrence of an infinite loop that might arise from a series of perception levels. In order to achieve this, Bennett introduces a comprehensive hypergame definition that encompasses all levels of strategies, perceptions, and strategy mappings.

Given that hypergames can effectively represent scenarios where different parties hold varying beliefs about the game, employing them to model conflicts is a logical progression. Bennett and Huxham pursued this using two phases, involving preliminary problem-solving and formal model-building and analysis [14]. The preliminary problem-solving phase aims to get a structured picture of the problem by identifying the players, interactions, and their respective objectives. The formal model-building and analysis phase takes this structured overview and considers the simplest possible theoretical systems of it. This process ensures not only the feasibility of constructing a model for the situation but also the inclusion of crucial information and known idealisations within this simplified framework. This simplified structure serves as a foundation for constructing more intricate models, wherein additional elements are incrementally introduced based on the outcomes of preliminary problem-solving. Each newly expanded model is then subjected to analysis.

Sasaki and Kijima extended this research by exploring Hyper Nash equilibria and introducing stable Hyper Nash equilibria [137], using a straightforward hypergame definition.

Definition 3.10 (Simple hypergame) A simple hypergame H is given by $(N, (G^i)_{i \in N})$, where:

- $N = \{1, \dots, n\}$ is a set of agents involved in the situation,

- $G^i = (N^i, S^i, u^i)$ is the subjective game of agent i , where:
 - N^i is a set of agents perceived by agent i .
 - $S^i = \times_{j \in N^i} S_j^i$ is a set of strategies perceived by agent i , where S_j^i is a set of strategies of agent j perceived by agent i .
 - $u^i = (u_j^i)_{j \in N^i}$ is a profile of utility functions perceived by agent i , where $u_j^i : S_j^i \rightarrow \mathbf{R}$ is agent j 's utility function perceived by agent i .

A Hyper Nash equilibrium combines the Nash equilibria from the individual games. It encompasses player one's Nash strategy as perceived by player one and player two's Nash strategy as perceived by player two. A set of Hyper Nash equilibria is $\mathbf{HN}(H) = \times_{i \in N} \mathbf{N}(G^i)_i$ of hypergame H .

Definition 3.11 (Hyper Nash equilibrium) There exists a Hyper Nash equilibrium, $(s_i^{i*})_{i \in N} \in \times_{j \in N} S_j^j$, of a hypergame H iff $\forall i \in N, s_i^{i*} \in \mathbf{N}(G^i)_i$.

Example 3.12 (The Fall of France Cont.) If we make 1 = Germans and 2 = Allies, we can create a hypergame;

$H = (N, (G^i)_{i \in N})$, $N = \{1, 2\}$, $G^1 = (N^1, S^1, u^1)$, $G^2 = (N^2, S^2, u^2)$ where $N^1 = \{2\}$, $S^1 = S_2^1 = \{RML, MN, N + C\}$, $N^2 = \{1\}$, $S^2 = S_1^2 = \{AML, AN\}$, with the matrices showing the ordinal utility functions. We can then calculate the Hyper Nash equilibria;
 $\mathbf{N}(G^1)_1 = \{AN, N + C\}$, $\mathbf{N}(G^2)_2 = \{AN, MN\}$,
 $\mathbf{HN}(H) = \mathbf{N}(G^1)_1 \times \mathbf{N}(G^2)_2 = \{AN, MN\}$

A stable Hyper Nash equilibrium occurs when the strategies of other players align with a player's perceptions. Therefore, the player has no incentive to update their perceptions, resulting in a 'stable' solution that remains robust even in repeated situations.

Definition 3.13 (Stable Hyper Nash equilibrium) There exists a stable Hyper Nash equilibrium, $(s_i^{i**})_{i \in N} \in \times_{j \in N} S_j^j$, of a hypergame H iff $\forall k \in N, (s_i^{i**})_{i \in N} \in \mathbf{N}(G^k)$.

The set of stable Hyper Nash equilibria is represented as $\mathbf{SHN}(H) = \cap_{i \in N} \mathbf{N}(G^i)$. Moreover, their research demonstrates that if a stable Hyper Nash equilibria exists, it is a Nash equilibria of the base game. A stable Hyper Nash equilibrium can only exist when Hyper Nash equilibria exist within the hypergame, and all of these Nash equilibria are a part of the base game.

Example 3.14 (The Fall of France Cont.) No stable Nash equilibria exist in this game as $\mathbf{SHN}(H) = \mathbf{N}(G^1) \cap \mathbf{N}(G^2) = \emptyset$. This means that the hypergame H is an unstable hypergame.

A level-0 (zero-level) hypergame denotes a scenario where there is no incomplete information; all players possess full knowledge of the game and are playing the same game. A level-1 (first-level) hypergame arises when one player has incomplete information or a

misperception about another player's preferences. In a level-2 (second-level) hypergame, a player is aware of the incomplete information of another player or aware that they are playing different games.

Hypergames serve as models for games with incomplete information, another such model is Bayesian games, as introduced by Harsanyi in 1967. Sasaki and Kijima explore the connection between these two models in their work [139]. They discuss both models and introduce a method to transform hypergames into a Bayesian game, creating a Bayesian representation. This transformation allows them to establish that the Hyper Nash equilibria of the hypergame and the Bayesian Nash equilibria of the Bayesian representation, lead to the same implications, as well as the best response equilibrium of extended hypergames and the Nash equilibrium of the Bayesian games.

Wang, Hipel, and Fraser [161] have contributed numerous solution concepts, including Nash stability, Metagame stabilities, Sequential stability, Stackelberg stability, and Limited-move stabilities. Their work shows that a hypergame will always have at least one overall solution if each individual game has one. Nash stability offers an outcome for a player when it represents their best achievable result, while the rest of the players' strategies are fixed, disregarding player perceptions. As outlined earlier, the overall outcome of the game is dependent on each player's perspective.

Players within hypergames benefit by learning or adapting to their opponents' strategies. One way this can be addressed is by using an Evolutionary approach. Instead of focusing on repeated games, a 'one-shot' hypergame is extended into a more dynamic form as demonstrated in [152]. A Network-type n-person hypergame is introduced which expresses a one-shot game that can represent the learning process in hypergame situations. Network-type Dynamic hypergames involve each player exchanging their outcomes and each player then improves their perceived strategy sets or payoff. This cycle continues until a termination criterion is met. Aggregated information and the principles of the two-person hypergame are shown to be crucial for effective learning.

Sasaki and Kijima proposed a new equilibrium concept called systems intelligent equilibrium (SIEq) for hypergames [138]. SIEq refines stable Hyper Nash equilibria (SHN) by considering some off-the-equilibrium plays. The relation between Nash equilibrium, SHN and SIEq was addressed and it was found that for a hypergame H , $SI(H) \subseteq SHN(H) \subseteq N(BG_H)$, where BG_H represents the base game of the hypergame. In some specific situations, the SIEq leads to Pareto-optimal outcomes from an objective viewpoint. Pareto-Optimisation occurs when an agent finds more than one Nash equilibrium, in this case, the agent chooses the Pareto-Optimal one (defined as a Pareto Nash equilibrium).

Hypergame theory can be extended to encompass both game-theoretic information and decision-theoretic information, within a table known as the hypergame normal-form introduced in [158]. Hypergame normal-form presents the payoff matrix in the lower right-hand corner with the belief of the game captured in the top rows. These belief models are represented using the Nash Equilibrium mixed strategies. The weighting of the contributing probability vectors is placed to the left of each row. In order to summarise the game, Hyperstrategies are used, which take the form of probability vectors encompassing all the available options. As well as introducing Hypergame Expected

Utility which accounts for outguessing in a hypergame context.

A new solution concept for hypergames is introduced in [136] called subjective rationalisability. This concept entails defining a hypergame model with a set of players and a set of games for each individual subjective game, from a set of viewpoints relevant to the situation. An action taken by the lowest agent in a viewpoint is called subjectively rationalisable for the viewpoint if it is the best response in the subjective game to some of the other agents. Moreover, this action should also be the best response within the subjective game of a viewpoint one step lower, and so on, for certain actions of other agents. An outcome in this situation is a combination of each agent's subjectively rationalisable action. However, it's important to note that subjective rationalisability is specifically a solution concept for one-shot hypergames. It cannot be employed if agents have the capability to modify their perspectives between decisions.

Conflicts can be categorised as instances of snapshot decision-making. In these scenarios, the equilibrium is a stable equilibrium [73]. Equilibria in hypergames can fall into two categories: hypergame-preserving equilibrium (where all players perceive the same solution) and hypergame-destroying equilibrium (not seen by at least one player). After establishing a hypergame, each player selects a strategy, and an overall equilibrium is established. This equilibrium is an element of the Cartesian product space of strategies. If this equilibrium is a hypergame-destroying equilibrium, then the equilibrium could be a persistent equilibrium if it's stable for all players. A snapshot equilibrium is the conflict in a snapshot decision-making problem, or a transitory equilibrium involving a dynamic decision or transfer to another phase.

General Meta Rationality (GMR) considers the potential responses of other players to prevent a given player's enhanced strategy from leading to a specific outcome. This allows the players to anticipate and factor in the possible responses of other players to an improvement by a player. GMR achieves stability through sequential sanctioning if there is more than one unilateral improvement for an outcome and all of the improvements are credibly blocked [1]. If a player can be deterred from making a unilateral improvement from the outcome due to an inescapable move, the outcome is termed Symmetric Meta Rational (SMR) [61]. In SMR outcomes, the other players have one joint sequential movement to respond to all the unilateral improvements of the player. This strategic coordination ensures that the player is always placed in a position that is either less preferred or equally preferred.

Swap Learning was introduced in [62] where players can update their own perception based on information contained within the actions of other players. One result proved that swap learning can only decrease misperception at the cost of experiencing inconsistencies. Therefore, a modified swap learning method was introduced, which yields constant beliefs and decreases misperception under certain conditions.

Chapter 4

Computability and Game Theory

Contents

4.1	Computability for Strategic Games	41
4.2	Computability for Sequential Games	42
4.3	Constructivism in Game Theory and Bounded Rationality	43
4.4	Applications of Games to Computability Theory	44

4.1 Computability for Strategic Games

Several concepts from non-cooperative game theory are investigated in order to find their Weihrauch degrees within [122]. The paper establishes the Weihrauch degree of a pure equilibrium and robust division, which accepts division by zero and returns an arbitrary value. While robust division is found to be slightly incomputable as it is reducible to deciding whether a real number is 0 or not. It is demonstrated by Pauly that every computable bi-matrix game has a computable Nash equilibrium. When the search for Nash equilibria is limited to games with a unique equilibrium, the problem becomes computable. The Weihrauch degree of the one-player Nash equilibrium was found to be equivalent to C_2^* , whereas the two-player results were proved to be equivalent to robust division in parallel. Additionally, it's shown that $\text{Nash}_1 < \text{Nash}_2$.

In the context of bi-matrix games, equilibria and their associated Weihrauch degrees are investigated in [124]. All-or-Unique Choice, a restriction of closed choice, is proven to be equivalent to robust division. The paper introduces a multivalued function BRoot , which maps polynomials to a root if one exists. It's demonstrated that $\text{AoUC} \equiv_W \text{BRoot}_2$, where BRoot_2 is just the task of solving $bx = a$.

An investigation into robust division is achieved within [86] where they discuss the strength of the computational problem. The paper examines both sequential and concurrent applications of robust division, demonstrating that it is not finitely concurrent. They found that three or more consecutive applications of powers of AoUC reduce to two. Furthermore, the paper establishes that invoking AoUC in parallel is Weihrauch reducible to invoking AoUC sequentially.

Las Vegas computable multivalued functions are introduced within [23]. The paper also presents the class of Las Vegas computable functions within the Weihrauch lattice, using probabilistic choice principles. Building on the fact that Nash is Weihrauch equivalent to $\text{AoUC}_{[0,1]}^*$, Brattka, Gheradi and Hölzl established that there is a Las Vegas algorithm for computing Nash equilibria. They also established that no fixed positive success probability is sufficient for robust division.

Monte Carlo computability extends the concept of Las Vegas computability. Las Vegas machines can recognise the failure of the advice in finite time, whereas Monte Carlo machines only recognise the failure in the limit. Monte Carlo computability and machines are defined using ordinary computable functions within [26]. It shows that Monte Carlo computable functions are closed under composition and that every Las Vegas computable function is Monte Carlo computable.

Bubelis showed in [35] that given any real algebraic number, there exists a three-person game with rational data which has a unique equilibrium point. The paper also introduces a method for reducing an arbitrary n -person game to a three-person one and shows that a three-player completely mixed game can be constructed with the equilibrium set being a manifold of dimension one.

4.2 Computability for Sequential Games

The exploration of a subclass of effective Gale-Stewart games which are recursively bounded and recursively presented games is conducted in [38]. The paper demonstrated how the set of winning strategies is a recursively bounded Π_1^0 -class. Additionally, they show that restricting recursively bounded Gale-Stewart games to polynomial time presented games have the same set of winning strategies as a recursively bounded, recursively presented game.

In the context of win/lose games with both closed and open outcomes, transitioning to two-player games with multiple outcomes introduces complexities to the task of finding Nash equilibria. This challenge occurs when attempting to employ finitely many uses of the Limited Principle of Omniscience (LPO) in parallel, as demonstrated in [95]. It also shows that for subgame-perfect equilibria, countably many uses of LPO become necessary.

The prisoner's dilemma is a widely examined aspect of game theory. This is a symmetric, two-player game with two strategies. The notion of repeating this game infinitely is achieved within [89] where they investigated different strategies and how computable they are. They discovered an inaccessible strategy which is the best response to a computable strategy with no computable best response.

The prisoner's dilemma is an example of a stage game, whose non-computable strategies were investigated within [109]. The paper's main theorem asserts that, under the assumptions regarding the stage game, a strategy computable by a Turing machine exists. However, no best response to this strategy can be implemented by a Turing machine. Moreover, the problem of finding a computable best response may not have

a computable solution, even if it's restricted to computable strategies that do admit a computable best response.

A sequential game involving three rounds with moves drawn from \mathbb{N} , as demonstrated by Rabin in [133], showcases a winning strategy for player two which was not computable. In a scenario where a two-player game is played repeatedly, with player one being aware that player two consistently employs an effectively computable strategy, it's shown that after a finite number of plays, player one can devise a method for ensuring consistent victory.

4.3 Constructivism in Game Theory and Bounded Rationality

The exploration of constructive aspects within game theory delving into instances and non-computability, and the overarching pursuit of a *more* constructive game theory, boast a rich historical lineage. Ever since Nash's groundbreaking contribution, Brouwer's Fixed Point Theorem has become integral to the bedrock of game theory. Paradoxically, Brouwer's fixed point theorem lacks a constructive proof. In fact, Orevkov [116] demonstrated its falsehood within the realm of Russian constructivism. The Weihrauch degree of Brouwer's fixed point theorem was classified in [27], aligning with the same classification as Weak König's Lemma.

The identical Weihrauch degree surfaces in various instances of non-computability within game theory. Examples include Gale-Stewart games lacking computable winning strategies [38, 97], and the observation that within the infinite repeated prisoner's dilemma, a computable strategy lacks a computable best-response (e.g., [89], [109]). What unites these examples is their focus on games of infinite duration, in contrast to the finite normal form games studied in this context.

Rabin presented a sequential game involving three rounds, where moves are drawn from \mathbb{N} . Remarkably, this game showcases a decidable determination of the winning player in any given play. Paradoxically, the second player possesses a winning strategy, yet this same player lacks any computable winning strategy [133]. While the Weihrauch degree inherent in this construction remains largely unexplored, Rabin's analysis inherently establishes it to be unequivocally higher than Weak König's Lemma.

The context of finite games in normal form was constructively explored by Bridges in [33]. This examination promptly highlighted that the minmax theorem cannot be established within this framework, given its reliance on the non-constructive principle LLPO. Bridges, along with coauthors, also delved into the task of constructing utility functions from preferences within a constructive framework, unearthing numerous challenges and obstacles [10, 32, 34].

Numerous authors have argued that, in the realm of game theory, the imperative for constructiveness is even more pronounced than in other branches of mathematics. This stems from the fact that the solution concepts of game theory explicitly stem from decision-making processes by agents, which are expected to adhere to the established principles of computability. As discussed in [124], the necessity of falsifiability inherently

places constraints on the degree of non-constructiveness that a scientific theory can embrace. For instance, while the Weihrauch degree of Weak König's Lemma aligns with falsifiability, it falls short of the heightened requirement we propose within game theory. Additional calls for a more constructivist stance in game theory have been articulated, as exemplified by Velupillai [159, 160].

4.4 Applications of Games to Computability Theory

4.4.1 Wadge Games

A Wadge game is defined as a two-player game with perfect information, where player one selects a natural number and player two responds by either choosing a natural number or passing [114]. For generalised Wadge games, when the lower cone forms a transparent cylinder, they characterise a lower cone within the Weihrauch degrees. Furthermore, it was found that if a Wadge game has a computable winning strategy, then it must also possess a computable point of discontinuity.

4.4.2 Reachability Games

The study of Weihrauch degrees of closed choice for finite sets and convex sets is completed within [87]. This examination enabled the consideration of scenarios where finite choice is reducible to some finite product of finite choice operators, using a reachability game. The winner of the game would inform them as to whether the reduction holds. A winning strategy for player one results in a witness for a non-reduction, whilst a winning strategy for player two results in witnesses for a reduction. This game was also implemented by Jones within [82].

4.4.3 Computational Complexity for Discrete Games

Hirschfeldt and Jockusch introduced a two-player reduction game denoted as $G(Q \rightarrow P)$ in their work [75]. They established that if $P \leq_W Q$, then player two possesses a winning strategy for the game $G(Q \rightarrow P)$; otherwise, player one possesses a winning strategy for the same game. This game is further expanded into a generalised reduction game, denoted as $\hat{G}(Q \rightarrow P)$, where the game terminates if either player lacks a legal move, resulting in a victory for the other player. They demonstrated that if $\text{RCA}_0 + Q \vdash P$, then player two possesses a winning strategy in the game $\hat{G}(Q \rightarrow P)$; conversely, if the implication does not hold, then player one possesses a winning strategy. These winning strategies correspond to implications and non-implications between Π_2^1 principles across ω -models of RCA_0 .

This work was extended to other formal systems by Dzhafarov, Hirschfeldt, and Reitzes within the context of [57]. They establish compactness, which demonstrates that if an implication $Q \rightarrow P$ between two principles holds, then there exists a winning strategy achieving victory in a number of moves bounded by a value independent of the specific game run. They illustrate how this framework leads to a novel type of analysis for

logical mathematical problems using Weihrauch reducibility. This comparison allows for a fine-structural examination of Π_2^1 principles encompassing both computability-theoretic and proof-theoretic aspects.

Part III

Computable Analysis and Game Theory

Chapter 5

The Non-computability of Nash Equilibria in Multiplayer Games

Contents

5.1	Introduction	49
5.2	Our Main Theorem	51
5.3	Roots of Polynomials	52
5.4	An Open Question and a Remark	70
5.5	Consequences of the Classification	72

This chapter is based on the paper ‘The Weihrauch Degree of Finding Nash Equilibria in Multiplayer Games’ which can be found [49]. This work is a collaboration with Arno Pauly.

5.1 Introduction

Is there an algorithm that reads games in strategic form and outputs some Nash equilibrium? This question is not only relevant for practical applications of Nash equilibria, whether in economics or computer science, but it also plays a central role in justifying Nash equilibrium as the outcome of rational behaviour. If an agent can discern their own strategy in a Nash equilibrium (in order to follow it), then all agents together ought to be able to compute a Nash equilibrium.

If the payoffs in our games are given as integers, the existence of algorithms to find Nash equilibria is readily verified. Here the decisive question is how efficient these algorithms can be. For two-player games, the problem is PPAD-complete [40], whereas the multiplayer variant is complete for FIXP [59], (the PPAD-completeness result from [54] is for ε -Nash equilibria, not for actual Nash equilibria). These complexity classifications are still a challenge for justifying Nash equilibrium as a solution concept since they are widely believed to be incompatible with the existence of efficient algorithms.

Our focus here is on payoffs given as real numbers. Essentially, this means that our algorithm has access to arbitrarily good approximations for the payoffs. However, an algorithm cannot confirm that two real inputs are equal – and it cannot even pick a true case between *the first input is not smaller* and *the second input is not smaller*. This is the non-constructive principle LLPO, which is easily seen to correspond to finding Nash equilibria in single-player games with just two options. We should not stop with this negative answer to our initial question, but instead, explore *how non-computable* the task of finding Nash equilibria is. As usual, this means identifying the degree of the problem for a suitable notion of reducibility. Here, this notion is Weihrauch reducibility.

Besides satisfying our curiosity, classifying the Weihrauch degree of finding Nash equilibria lets us draw some interesting conclusions. For example, there is a Las Vegas algorithm to compute Nash equilibria, but we cannot provide any lower bound for its success rate. We will discuss these consequences further in Section 5.5. For games with one or two players, a complete classification has already been obtained in [122], but the situation for multiplayer games remained open and will be addressed here.

We use the well-known ‘algorithm’ called Cylindrical Algebraic Decomposition (CAD) with a few modifications to reach our results. The modifications are necessary because in its original form, CAD assumes the equality of coefficients to be decidable. We explore the computable content of CAD by investigating each aspect of the algorithm to what extent they are computable when working with real numbers. The obstacles can be overcome by moving to suitable over-approximations.

Computability in the countable, discrete realm may be the better-known concept, through the notion of Turing computability. There is also the algebraic approach to computability as put forth by Blum, Shub, and Smale [17]. That model does not fit the justification for why computability is required in game theory. Nash equilibria still are not computable in the BSS-model though [123] works perfectly well on most spaces of interest of cardinality up to the continuum. The field that delves into the study of computability in such settings is referred to as *computable analysis*, whose introduction can be seen in Chapter 2.

Turing machines inherently possess infinite tapes, allowing us to employ infinite binary sequences as their inputs and outputs. Computations then no longer halt but instead continue to produce more and more output. To get computability for interesting objects such as the reals, we code them via the infinite binary sequences. This yields the notion of a *represented space*. In the case of the reals, an encoding based on the decimal or binary expansion would yield an unsatisfactory notion of computability (as multiplication by 3 would not be computable). However, an encoding via sequences of rational numbers converging with a known rate (e.g. we could demand that $|q_n - x| < 2^{-n}$, where x is coded real and q_n the n -th approximation) works very well [155]. Essentially, this approach renders all naturally occurring continuous functions computable. The standard representation of the real numbers is also consistent with assuming that real numbers are obtained by repeating physical measurements over and over and thus obtaining higher and higher expected accuracy [121].

5.2 Our Main Theorem

Once we have established the method for representing real numbers, it is trivial to obtain a representation for finite games in strategic form and a representation for mixed strategy profiles. We can then define the multivalued function Nash mapping finite games in strategic form to some Nash equilibrium. We refer to the restriction of Nash to two-player games as Nash₂. Our main goal is to classify the Weihrauch degree of Nash. We do this by comparing it to a benchmark principle called All-Or-Unique Choice, AoUC_[0,1] (Definition 2.21).

Definition 5.1 AoUC_[0,1] receives an abstract input that expresses which $x \in [0, 1]$ are valid solutions as follows: Initially, all of $[0, 1]$ is a valid solution. This could remain the case forever (the *all*-case), or at some point, we receive the information that there is just a unique correct answer, and we are told what that one is (the *unique*-case).

We refer to AoUC_[0,1]^{*} and AoUC_[0,1][◇] within this section which describes AoUC_[0,1] being invoked finitely many times in parallel and invoked any finitely many number of times, respectively (Definition 2.16).

Definition 5.2 Let BRoot : $\mathbb{R}[X] \rightrightarrows [0, 1]$ map real polynomials to a root in $[0, 1]$, provided there is one, and to arbitrary $x \in [0, 1]$ otherwise. Let BRoot_k (BRoot_{≤k}) be the restriction of BRoot to polynomials of degree (less than -or-equal to) k .

Another highly relevant multivalued function for us is BRoot. Let BRoot_k (BRoot_{≤k}) be the restriction of BRoot of polynomial of degree (less-or-equal than) k . In particular, we see that BRoot_{≤1} is just the task of solving $bx = a$. The obstacle for this is that we do not know whether $b = 0$ and anything $x \in [0, 1]$ is a solution, or whether $b \neq 0$ and we need to answer $\frac{a}{b}$. Following an observation by Brattka, it was shown as [86, Proposition 8] that AoUC_[0,1] \equiv_W BRoot_{≤1}. The main result from [122] is Nash₂ \equiv_W AoUC_[0,1]^{*}. The remaining ingredient of our first main result is found in Corollary 5.31 in Subsection 5.3.3:

Theorem 5.3 AoUC_[0,1]^{*} \leq_W Nash \leq_W AoUC_[0,1][◇]

As established in [86] AoUC_[0,1]^{*} $<_W$ AoUC_[0,1][◇], so at least one of the two reductions in Theorem 5.3 is strict. Furthermore, from the outcomes presented in [86] it also follows that AoUC_[0,1][◇] \equiv_W AoUC_[0,1]^{*} \star AoUC_[0,1]^{*}, where $f \star g$ lets us first apply g , then do some computation, and then apply f . Consequently, any finite number of oracle calls to AoUC_[0,1] can be rearranged to happen in two phases, where all calls within one phase are independent of each other. Drawing upon the outcome in [122], we can conclude that from a multiplayer game G we can compute a two-player game G' , take any Nash equilibrium of G' , and compute another two-player game G'' from that such that given any Nash equilibrium to G'' we could compute a Nash equilibrium for G . It seems very plausible to us that AoUC_[0,1]^{*} \equiv_W Nash should hold, but constructing a proof for this equivalence has posed a challenge. Thankfully, the outcomes discussed in Section 5.5, stemming from Theorem 5.3, do not hinge on the resolution of these specific details.

The lower bound in Theorem 5.3 clearly already follows from $\text{AoUC}_{[0,1]}^* \equiv_{\text{W}} \text{Nash}_2$. For the upper bound, we bring three ingredients together. The first result shows that a preprocessing step that identifies a potential support for a Nash equilibrium can be absorbed into the Weihrauch degree $\text{AoUC}_{[0,1]}^*$. Once we know the support we are going to use, we have a solvable system of polynomial inequalities whose solutions are all Nash equilibria.

We are thus led to investigate the Weihrauch degrees pertaining to solving systems of polynomial (in)equalities. We completely classify the degree of finding (individual) roots within given bounds of finitely many (univariate) polynomials:

Theorem 5.4 (Proven as Corollary 5.8 below) $\text{AoUC}_{[0,1]}^* \equiv_{\text{W}} \text{BRoot}^*$

Consequently, disregarding the precise count of required oracle calls, the task of finding polynomial roots does not pose a greater challenge than solving equations in the form $bx = a$. The proof of Theorem 5.4 in turn makes use of a result from [96] on finding zeros of real functions with finitely many local minima.

To prove Theorem 5.3 we need more, namely we need to solve systems of (in)equalities for multivariate polynomials. This aspect is addressed in the forthcoming Corollary 5.30. Through an examination of cylindrical algebraic decomposition, augmented with minor adaptations, we demonstrate that access to $\text{AoUC}_{[0,1]}^*$ lets us compute finitely many candidate solutions including a valid one. A comprehensive introduction to cylindrical algebraic decomposition (CAD) is available in [81]. CAD involves the partitioning of \mathbb{R}^n into semi-algebraic cylindrical cells.

5.3 Roots of Polynomials

We consider the computability aspects of finding the roots of polynomials. Our exploration of polynomial root finding unfolds across three distinct scenarios. In Subsection 5.3.1 we look into monic univariate polynomials. In Subsection 5.3.2 we drop the restriction to monic, and in Subsection 5.3.3 we handle multivariate polynomials.

In order to conceptualise the task of polynomial root finding, it becomes essential to establish a clear methodology for representing polynomials. A polynomial is represented by providing an upper bound to its degree, plus a tuple of real numbers constituting all relevant coefficients. For example, we could be given the same polynomial as either $0x^2 + 3x - 5$ or $3x - 5$. If we were demanding to know the exact degree, we could no longer compute the multiplication and addition of polynomials, which would be clearly unsatisfactory. This matter is discussed in detail in [131, Section 3]. We denote the represented space of real univariate polynomials as $\mathbb{R}[X]$ and the space of real multivariate polynomials as $\mathbb{R}[X^*]$. For the latter, we assume that each polynomial comes with the exact information of what finite set of variables it refers to. Both $\mathbb{R}[X]$ and $\mathbb{R}[X^*]$ are coPolish spaces, see [141, 31, 36].

5.3.1 Monic univariate polynomials

It is known since the dawn of computability theory that the fundamental theorem of algebra is constructive, more precisely, that given a monic polynomial with real (or complex) coefficients, we can compute the unordered tuple of its complex roots, each repeated according to its multiplicity. The latter formulation was established by Specker [146].

Of course, we cannot decide which of the complex roots are real. The task of selecting a real number from a k -tuple of complex numbers containing at least one real number is Weihrauch equivalent to C_k (Definition 2.20). A similar task equivalent to C_k is to identify a 0-entry in a k -tuple of real numbers containing at least one 0. Given real numbers $\varepsilon_0, \varepsilon_1, \dots, \varepsilon_{k-1}$ we can construct the monic polynomial $\prod_{i < k} ((x - \frac{i}{k})^2 + |\varepsilon_i|)$, which will have a root at $\frac{i}{k}$ iff $\varepsilon_i = 0$. This shows that finding real roots of monic polynomials is Weihrauch equivalent to $C_2^* = \bigsqcup_{k \in \mathbb{N}} C_k$.

5.3.2 Univariate Polynomials

In general, we do not know the degree of a polynomial and thus cannot restrict ourselves to the monic case for root-finding. It has been shown by Le Roux and Pauly [96] that knowing a finite upper bound k on the number of local extrema lets us find a root of a continuous function (if it has one in a bounded interval) using $C_{\{1, \dots, 3^k\}}$. Essentially, this observation suggests that for polynomial root finding in $[0, 1]$, knowing the precise degree versus knowing an upper bound makes only a quantitative, but not a qualitative difference – if we exclude the zero polynomial!

While it may seem counterintuitive that it should be the zero polynomial that makes root finding more difficult, we will see that this is the case, and explore how much.

We have defined BRoot to be a total map. If the input is a polynomial without a root in the unit interval, it will return an arbitrary element of the unit interval instead. However, the restriction of BRoot which is defined only on polynomials with a root in the unit interval is in fact equivalent to BRoot itself. It's worth noting that this is a special case of Lemma 5.11 which we will prove later.

Proposition 5.5. $\text{BRoot}_{\leq 2k+1} \leq_W \text{AoUC}_{[0,1]} \times C_{3^k}$

Proof. A polynomial p of degree at most $2k + 1$ is either the 0 polynomial, or has at most k local minima. If it is not 0, we will recognize this, and moreover, can find some $a \leq 0, b \geq 1$ with $p(a) \neq 0 \neq p(b)$. In this case, [96, Theorem 4.1] applies and lets us compute a 3^k -tuple of real numbers amongst which all zeros of p between a and b will occur.

We recall that $\text{AoUC}_{[0,1]} \equiv_W \text{AoUC}_{[a,b]^n}$ [86, Corollary 12]. We let the input to $\text{AoUC}_{[a,b]^n}$ be $[a, b]^n$ as long as $p = 0$ is consistent, and if we learn that $p \neq 0$, we collapse the interval to the 3^k -tuple we obtain from [96, Theorem 4.1]. The input to C_{3^k} is initially $\{0, \dots, 3^k - 1\}$. We only remove elements after we have confirmed $p \neq 0$, and then we remove j if the j -th candidate obtained from [96, Theorem 4.1] is not a root of p , or falls outside of $[0, 1]$.

To obtain the answer to $\text{BRoot}_{\leq 2k+1}$, we just use the answer from C_{3^k} to indicate the answer of which component of the tuple provided by $\text{AoUC}_{[a,b]^n}$ to use as the final output. \square

The following is also a direct consequence of [87, Theorem 12], but its proof is much more elementary; and a direct consequence of [23, Proposition 17.4], but again with a simpler proof:

Proposition 5.6. $\text{AoUC}_{[0,1]} \times C_2 \not\leq_W \text{BRoot}$

Proof. Assume that the reduction would hold. We may assume that elements of $\mathcal{A}([0,1])$ are represented via enumerations of open intervals with rational endpoints exhausting their complement. Fix some standard name q for $[0,1] \in \mathcal{A}([0,1])$, which means that q just never enumerates any interval (without ever providing positive information that no interval is going to be enumerated in the future). Assume further that there is some name r for an input to C_2 such that (q,r) gets mapped to a non-zero polynomial p by the inner reduction witness. Since p is ensured to be non-zero already by a finite prefix of its names, and since the inner reduction witness is continuous, it holds that there is an $M \in \mathbb{N}$ such that any (q',r) with $d(q,q') < 2^{-M}$ gets mapped to a non-zero polynomial. Restricting $\text{AoUC}_{[0,1]}$ to names from $\{q' \mid d(q,q') < 2^{-M}\}$ does not change its Weihrauch degree (as q provides no information at all). By [96, Corollary 4.3], if we exclude the 0 polynomial from the domain of BRoot , then C_2^* suffices to find a root. We can thus conclude $\text{AoUC}_{[0,1]} \leq_W C_2^*$, but this contradicts [122, Theorem 22].

Thus, it would need to hold that each pair (q,r) gets mapped to the 0 polynomial. But since answering constant 0 is a computable solution to $\text{BRoot}(0)$, this, in turn, would imply that C_2 is computable, which is absurd. We have thus arrived at the desired contradiction. \square

While our preceding proposition shows the limitations of BRoot for solving multiple non-computable tasks in parallel, the following result from the literature reveals that the slightly more complex nature of $\text{AoUC}_{[0,1]}$ is central. Note that $C_n \times C_m \leq_W C_{n \cdot m}$, hence C_k has an inherently parallel nature for $k \geq 4$.

Proposition 5.7 ([96, Proposition 4.6]). $C_k \leq_W \text{BRoot}_{2^k}$

Corollary 5.8 $\text{AoUC}_{[0,1]} <_W \text{BRoot} <_W \text{AoUC}_{[0,1]}^*$

Proof. The first reduction follows from $\text{AoUC}_{[0,1]} \equiv_W \text{BRoot}_{\leq 1}$ ([86, Proposition 8]). That it is strict comes from [96, Proposition 4.6] showing that otherwise, we would have $C_3 \leq_W \text{AoUC}_{[0,1]} \leq_W \text{LPO}$, contradicting a core result from [163].

The second reduction follows from Proposition 5.5, together with $C_{k+1} \leq_W C_2^k$ and $C_2 \leq_W \text{AoUC}_{[0,1]}$ (both from [122]). Its strictness is a consequence of Proposition 5.6. \square

5.3.3 Multivariate Polynomials and Cylindrical Algebraic Decomposition

Our focus now shifts towards multivariate polynomials. Within this domain, two intriguing problems arise: the pursuit of a root for an individual multivariate polynomial and the search for a solution to a system of polynomial inequalities (restricting ourselves to the non-strict case for now). In both scenarios, our emphasis rests on the bounded case, given its relevance to Nash equilibria. Additionally, the unbounded case promptly leads us to LPO*.

Definition 5.9 Let $\text{BMRoot} : \mathbb{R}[X^*] \rightrightarrows [0, 1]^*$ map real polynomials to a root in $[0, 1]^*$, provided there is one; and to an arbitrary $x \in [0, 1]^*$ otherwise.

Our definition of BMRoot extends to all polynomials, and when confronted with a lack of roots within the unit hypercube, it provides an arbitrary element from the unit hypercube as an output. The latter scenario bears lesser significance, Lemma 5.11 detailed below shows that restricting BMRoot to polynomials having roots in the unit hypercube does not change its Weihrauch degree.

Definition 5.10 Let $\text{BPIneq} : (\mathbb{R}[X^*])^* \rightrightarrows [0, 1]^*$ map a sequence of polynomials P_1, \dots, P_k to a point $x \in [0, 1]^*$ such that $\forall i \leq k, P_i(x) \geq 0$ if such a point exists, and to any $x \in [0, 1]^*$ otherwise.

Lemma 5.11 There is a computable multivalued map $\text{MakeZero} : \mathbb{R}[X^*] \rightrightarrows \mathbb{R}[X^*]$ such that whenever $g \in \text{MakeZero}(f)$, then g has a zero in the unit hypercube; and if f already had a zero in the unit hypercube, then $f = g$.

Proof. Given an n -variate polynomial f , we can compute $c := \min\{|f(x)| \mid x \in [0, 1]^n\}$ since the unit hypercube is computably compact and computably overt. Let \mathbb{T} be Plotkin's T , i.e. the space with the truth values 0, 1 and undefined (\perp). We can compute $\text{sign}(f(0^n)) \in \mathbb{T}$ (where we consider the sign of 0 to be undefined).

The operation $\text{Merge} : \subseteq \mathbb{T} \times \mathbf{X} \times \mathbf{X} \rightarrow \mathbf{X}$ is defined on all inputs except (\perp, x, y) for $x \neq y$, and satisfies $\text{Merge}(0, x, y) = x$, $\text{Merge}(1, x, y) = y$ and $\text{Merge}(\perp, x, y) = x = y$. It is easy to see that Merge is computable for $\mathbf{X} = \mathbb{R}$. We use it to compute $\bar{c} = \text{Merge}(\text{sign}(f(0^n)), c, -c)$ and find that $f - \bar{c}$ meets the requirements to be an output to $\text{MakeZero}(f)$. Essentially, we just shift f vertically by the minimal amount required to make it have a zero in the unit hypercube. \square

Proposition 5.12. $\text{BMRoot} \equiv_{\text{W}} \text{BMRoot}^*$.

Proof. It suffices to show that $\text{BMRoot}^2 : \mathbb{R}[X^*] \times \mathbb{R}[X^*] \rightrightarrows [0, 1]^* \times [0, 1]^*$ is reducible to the map $\text{BMRoot} : \mathbb{R}[X^*] \rightrightarrows [0, 1]^*$. We are given two multivariate polynomials P and Q . We rename variables, in order to ensure that each polynomial uses different variables. By Lemma 5.11 we can assume w.l.o.g. that P and Q each have a root in the unit hypercube.

We then apply BMRoot to $P(\bar{x})^2 + Q(\bar{y})^2$ and obtain a root (\bar{x}_0, \bar{y}_0) of the latter polynomial. Now, \bar{x}_0 is a root of P and \bar{y}_0 is a root of Q .

□

We will conclude that $\text{AoUC}_{[0,1]}^\circ$ is an upper bound for the Weihrauch degree of BPineq (and thus BMRoot) as a corollary of the main theorem of this subsection, Theorem 5.29. The way we obtain Theorem 5.29 is via an analysis of how constructive CAD is if we do not take the equality test on the reals for granted.

To tackle the task of finding a finite over approximation of the roots in multivariate polynomials, even in the absence of prior knowledge about the degrees of these polynomials, we implement the CAD algorithm. This approach enables us to systematically deconstruct multivariate polynomials into lower-variate forms. This can be repeated until we have a set of univariate polynomials. Subsequently, we can find the roots of these univariate polynomials in the same way as Section 5.3.2. This establishes a foundation for determining the set of solutions within a system of polynomial equations and inequalities. The subsequent step involves ‘lifting’ these univariate polynomials to the original multivariate setting to pinpoint their roots within the context of the overarching problem.

The CAD algorithm has three phases; projection, base, and extension. This algorithm is driven by an input, represented as a set \mathcal{F} of n -variate polynomials. In the projection phase, a sequence of $n - 1$ steps is executed, each resulting in the creation of new polynomial sets. The zero set of the resulting polynomials consists of the projection of the significant points. The base phase isolates the real roots of the univariate polynomials from the outputs of the projection phase. Each root and one point in the intervals between roots are then used as sample points in the decomposition of \mathbb{R}^1 . The extension phase constructs sample points for all regions of the CAD of \mathbb{R}^n . This phase also consists of $n - 1$ steps which takes the sample points from the base phase and ‘lifts’ them into \mathbb{R}^2 for each region in the stack. This is repeated until we have sample points to all regions of the CAD of \mathbb{R}^n .

Our version of the CAD algorithm always outputs the maximum amount of sample points the polynomials could have, regardless of the unknown coefficients or degree. This does not affect the standard CAD algorithm, as the sample points are always either from the root or within the intervals between the roots; therefore, we can end up with extra sample points.

Definition 5.13 • A region \mathcal{R} is a connected subset of \mathbb{R}^n .

- The set $Z(\mathcal{R}) = \mathcal{R} \times \mathbb{R} = \{(\alpha, x) \mid \alpha \in \mathcal{R}, x \in \mathbb{R}\}$ is called a cylinder over \mathcal{R} .
- Let f, f_1, f_2 be continuous, real-valued functions on \mathcal{R} . A f -section of $Z(\mathcal{R})$ is the set $\{(\alpha, f(\alpha)) \mid \alpha \in \mathcal{R}\}$ and a (f_1, f_2) -sector of $Z(\mathcal{R})$ is the set $\{(\alpha, \beta) \mid \alpha \in \mathcal{R}, f_1(\alpha) < \beta < f_2(\alpha)\}$.

Sections and sectors can both be referred to as cells when discussing them as a whole group. Within the context of CAD, the regions from \mathcal{R} that make an appearance signify the locations where the $n + 1$ -variate polynomial possesses a root (with the first n -variables ranging over \mathcal{R}). Specifically, the first n variables range over \mathcal{R} in this process.

Concurrently, the sectors encapsulate the intermediary segments where the polynomial remains consistently positive or negative. This contributes to a decomposition, which entails the fragmentation of a given region into smaller, distinct components. Therefore, a decomposition of \mathbb{R} is a set of regions, defined above as sectors and sections.

Definition 5.14 Let $\mathcal{R} \subseteq \mathbb{R}^n$. A decomposition of \mathcal{R} is a finite collection of disjoint regions (components) whose union is \mathcal{R} : $\mathcal{R} = \bigcup_{i=1}^k \mathcal{R}_i$, $\mathcal{R}_i \cap \mathcal{R}_j = \emptyset$ whenever $i \neq j$.

A stack over \mathcal{R} is a decomposition of $\mathcal{R} \times \mathbb{R}$ comprising a combination of f_i -sections and (f_i, f_{i+1}) -sectors, where $f_0 < \dots < f_{k+1}$ for all $x \in \mathcal{R}$ and $f_0 = -\infty, f_{k+1} = +\infty$.

The stack decomposition of $\mathbb{R}^0 = \{0\}$ is $\{\{0\}\}$. A stack decomposition of \mathbb{R}^{n+1} is a decomposition of the form $\bigcup_{\mathcal{R} \in \mathfrak{D}} S_{\mathcal{R}}$, where each $S_{\mathcal{R}}$ is a stack over \mathcal{R} , and \mathfrak{D} is a stack decomposition of \mathbb{R}^n .

The initial phase revolves around projecting polynomials from n variables to a set in $n - 1$ variables. Within this process, for each section i , a real polynomial $f \in \mathbb{R}[x_1, \dots, x_{n-1}][x_n]$ in n -variables can be deconstructed into the coefficients for every power of x : $f(x_1, \dots, x_{n-1}, x_n) = \sum_{j=0}^d f^j(x_1, \dots, x_{n-1})x_n^j$, where d signifies the upper bound for the degree of the polynomial.

We are interested in the coefficient of the highest power of a greatest common divisor (gcd) as the number of common zeros of two polynomials, $f, g \in k[x]$, is $\deg(\gcd(f, g))$, [81, Lemma 4.1]. Also, the number of distinct zeros of $f \in k[x]$ is $\deg(f) - \deg(\gcd(f, f'))$, [81, Lemma 4.2]. The principal subresultant coefficient (psc) allows us to know the degree of the gcd of two polynomials.

Definition 5.15 Let R be a ring and $f, g \in R[x]$ with $\deg(f) = m, \deg(g) = n, m \geq n$. The k^{th} principal subresultant coefficient of f and g is

$$\text{psc}_k(f, g) = \det(M_k^{(k)}), \quad 0 \leq k \leq n$$

where M_0 is the Sylvester matrix of f and g , and then M_k is obtained by deleting certain rows and columns from M_0 .

The Sylvester matrix is a matrix of multivariate polynomials within this situation. The process of finding the determinant of a matrix where each entry can be a multivariate polynomial remains the same as finding the determinant for any matrix. However, the determinant will be a polynomial expression involving the variables of the polynomials within the original matrix. Therefore, if there are two multivariate polynomials with the variables x_1, x_2, \dots, x_n , the determinant will be a polynomial involving all the variables x_1, x_2, \dots, x_n . For the psc we require the knowledge of Sylvester Matrices and how M_k is created.

Definition 5.16 For two polynomials $f_1(x) = a_m x^m + \dots + a_0$ and $f_2(x) = b_n x^n + \dots + b_0$ of degrees m and n respectively, the Sylvester matrix is an $(m + n) \times (m + n)$ matrix formed by filling the matrix beginning with the upper left corner with the coefficients of $f_1(x)$. Then, shifting down one row and one column to the right and filling in the

coefficients starting there until they hit the right side. This process is then repeated for the coefficients of $f_2(x)$.

Example 5.17 Let $f_1 = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ and $f_2 = b_2x^2 + b_1x + b_0$ where $m = 4$ and $n = 2$. Therefore, M_0 is a 6×6 matrix filled with the coefficients of f_1 and f_2 .

$$M_0 = \begin{bmatrix} a_4 & a_3 & a_2 & a_1 & a_0 & \\ & a_4 & a_3 & a_2 & a_1 & a_0 \\ b_2 & b_1 & b_0 & & & \\ & b_2 & b_1 & b_0 & & \\ & & b_2 & b_1 & b_0 & \\ & & & b_2 & b_1 & b_0 \end{bmatrix}$$

M_1 is made from deleting one row from the f_1 and f_2 coefficients as well as one column. A divider is placed in the matrix after the $(m + n - 3)$ -column. This allows for multiple columns to be used to ensure the matrix is a square, shown as $M_k^{(l)}$.

$$M_1 = \left[\begin{array}{ccc|cc} a_4 & a_3 & a_2 & a_1 & a_0 \\ b_2 & b_1 & b_0 & & \\ & b_2 & b_1 & b_0 & \\ & & b_2 & b_1 & b_0 \end{array} \right]$$

Using $M_k^{(l)}$, we have $k = 1$ and l is equal to the column used to guarantee a square matrix.

$$M_1^{(4)} = \begin{bmatrix} a_4 & a_3 & a_2 & a_1 \\ b_2 & b_1 & b_0 & \\ & b_2 & b_1 & b_0 \\ & & b_2 & b_1 \end{bmatrix} \quad M_1^{(5)} = \begin{bmatrix} a_4 & a_3 & a_2 & a_0 \\ b_2 & b_1 & b_0 & \\ & b_2 & b_1 & \\ & & b_2 & b_0 \end{bmatrix}$$

Definition 5.18 The *reductum*, $\hat{f}_i^{k_i}$ of a polynomial f_i is

$$\hat{f}_i^{k_i}(x_1, \dots, x_{n-1}, x_n) = \sum_{j=0}^{k_i} f_i^j(x_1, \dots, x_{n-1})x_n^j$$

where $0 \leq k_i \leq d_i$. Here, f_i is a multivariate polynomial in x_1, \dots, x_n and $\hat{f}_i^{k_i}$ is a univariate polynomial with multivariate coefficients existing of (x_1, \dots, x_{n-1}) .

- Lemma 5.19**
1. The reductum of polynomials is computable.
 2. The derivative of a polynomial is computable.
 3. Given two polynomials p, q , we can compute a finite tuple where every well-defined psc appears within the tuple.

- Proof.*
1. Calculating the reductum requires the rearrangement of the polynomial $\hat{f}_i^{k_i}$, in order to group all of the coefficients for every power of $x_n, x_n^{k_i}, \dots, x_n^0$. This is trivially computable, as no tests on the coefficients need to be performed.
 2. Calculating the derivative of a polynomial merely requires the multiplication of coefficients with natural numbers.
 3. As we do not have access to the exact degrees of the polynomials, but merely to some upper bound, we do not even know which of the psc's are well-defined. Let n be the upper bound of $\deg(p)$ and m be the upper bound of $\deg(q)$. We have a finite potential number of pairs of polynomials, $(n + 1) \times (m + 1)$ combinations. These can be used to calculate the psc, under the assumption that the current degree of the pair of polynomials are the correct degrees.

□

We point out that the use of an ‘overapproximation’ in Lemma 5.19 is unavoidable. If we knew how many principal subresultant coefficients there are for f and g , we would know the degree of g .

Definition 5.20 Let $\mathcal{F} = \{f_1, f_2, \dots, f_r\} \subset \mathbb{R}[x_1, \dots, x_{n-1}][x_n]$ be a set of multivariate real polynomials and the region \mathcal{R} . We say that \mathcal{F} is delineable on \mathcal{R} if it satisfies the following invariant properties:

1. For every $1 \leq i \leq r$, the total number of complex roots of $f_i(y)$ remains invariant as y varies over \mathcal{R} .
2. For every $1 \leq i \leq r$, the number of distinct complex roots of $f_i(y)$ remains invariant as y varies over \mathcal{R} .
3. For every $1 \leq i \leq j \leq r$, the total number of common complex roots of $f_i(y)$ and $f_j(y)$ remains invariant as y varies over \mathcal{R} .

Given a set of polynomials $\mathcal{F} \subset \mathbb{R}[x_1, \dots, x_{n-1}][x_n]$ we can compute another set of $(n - 1)$ -variate polynomials $\text{proj}(\mathcal{F}) \subset \mathbb{R}[x_1, \dots, x_{n-1}]$, which characterises the maximal connected \mathcal{F} -delineable sets of \mathbb{R}^{n-1} . The projection can be summarised by

$$\mathcal{F}_0 = \mathcal{F} \subset \mathbb{R}[x_1, \dots, x_{n-1}, x_n], \mathcal{F}_1 \subset \mathbb{R}[x_1, \dots, x_{n-1}], \dots, \mathcal{F}_{n-1} \subset \mathbb{R}[x_1]$$

Definition 5.21 Given a set of polynomials $\mathcal{F} \subset \mathbb{R}[x_1, \dots, x_{n-1}][x_n]$ we can compute another set of $(n - 1)$ -variate polynomials $\text{proj}(\mathcal{F}) \subset \mathbb{R}[x_1, \dots, x_{n-1}]$ which characterises the maximal connected \mathcal{F} -delineable subsets of \mathbb{R}^{n-1} .

Let $\mathcal{F} = \{f_1, f_2, \dots, f_r\}$ then

$$\text{proj}(\mathcal{F}) = \text{proj}_1(\mathcal{F}) \cup \text{proj}_2(\mathcal{F}) \cup \text{proj}_3(\mathcal{F})$$

where

$$\begin{aligned} \text{proj}_1 &= \{f_i^k \mid 1 \leq i \leq r, 0 \leq k \leq d_i\}, \\ \text{proj}_2 &= \{\text{psc}_l^{x_n}(\hat{f}_i^k, D_{x_n}(\hat{f}_i^k)) \mid 1 \leq i \leq r, 0 \leq l < k \leq d_i - 1\}, \end{aligned}$$

$$\text{proj}_3 = \{\text{psc}_m^{x_n}(\hat{f}_i^{k_i}, \hat{f}_j^{k_j}) \mid 1 \leq i < j \leq r, 0 \leq m \leq k_i \leq d_i, 0 \leq m \leq k_j \leq d_j\}.$$

We have a multivariate polynomial over \mathbb{R} in terms of x_1, \dots, x_n . The $\text{psc}_i^{x_n}$ denotes the i^{th} principle resultant coefficient w.r.t x_n . This is read as a univariate polynomial in x_n with the coefficients being multivariate polynomials in x_1, \dots, x_{n-1} . D_{x_n} denotes the formal derivative operator with respect to x_n

Using Definition 5.20 the projections have invariant properties. For $\text{proj}_1(\mathcal{F})$ this implies that the degree w.r.t x_n of the polynomials is constant over \mathcal{R} if $\text{proj}(\mathcal{F})$ is invariant over $\mathcal{R} \subset \mathbb{R}^{n-1}$ and hence the number of roots of each polynomial is constant over \mathcal{R} . For $\text{proj}_2(\mathcal{F})$ it implies that the gcd of each polynomial and its derivative has a constant degree ([81, Lemma 4.4], [107, Corollary 7.7.9]). This in combination with $\text{proj}_1(\mathcal{F})$ shows that the number of distinct zeros of each polynomial in \mathcal{F} is constant [81, Lemma 4.2].

When $\text{proj}_3(\mathcal{F})$ is considered along with $\text{proj}_1(\mathcal{F})$ the invariant property implies that the number of common zeros of each pair of polynomials in \mathcal{F} is constant [81, Lemma 4.4]. The set $\text{proj}(\mathcal{F})$ characterises the sets over which there are a constant number of real zeros of the polynomial in \mathcal{F} ([81, Lemma 4.5], [107, Lemma 8.6.3]).

The algorithm involves the upward ‘lifting’ of univariate polynomials from \mathbb{R}^{i-1} to \mathbb{R}^i . This elevation is achieved by evaluating the polynomials in $\text{proj}^{n-i+1}(\mathcal{F})$ over a sample point α . This results in a set of univariate polynomials in x_i corresponding to the values of $\text{proj}^{n-i+1}(\mathcal{F})$ on the ‘vertical’ line $x_{i-1} = \alpha$. These univariate polynomials are treated the same in each ‘lifting’ phase until they reach \mathbb{R}^n .

In order to help with the concept of the CAD algorithm and the projections, an example is provided below.

Example 5.22 An intuitive example of the CAD algorithm using the polynomial $ay^2 - bx^3 - cx^2$. First we will look at the polynomial with $a, b = 1, c = 0$.

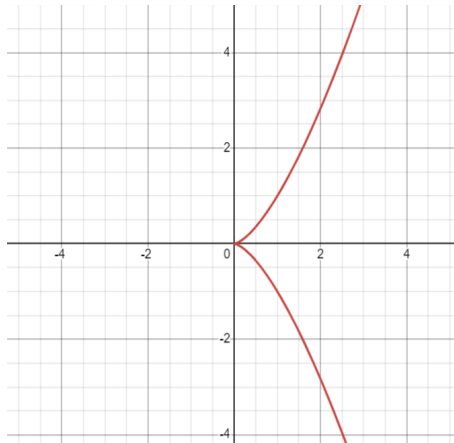


Figure 5.1: Graph of $ay^2 - bx^3 - cx^2$ when $a, b = 1, c = 0$

Here there is one root $x = 0$. Hence the proj to \mathbb{R}^1 consists of 3 cells $x < 0$, $x = 0$, $x > 0$.

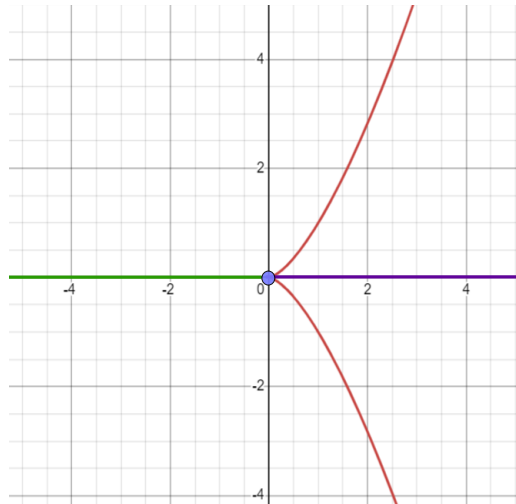


Figure 5.2: Projection onto the x-axis

In order to expand this to the \mathbb{R}^2 we check each cell of the projection individually. Lets start with $x < 0$. This is very simple as there is nothing set in the y-axis, therefore y is free.

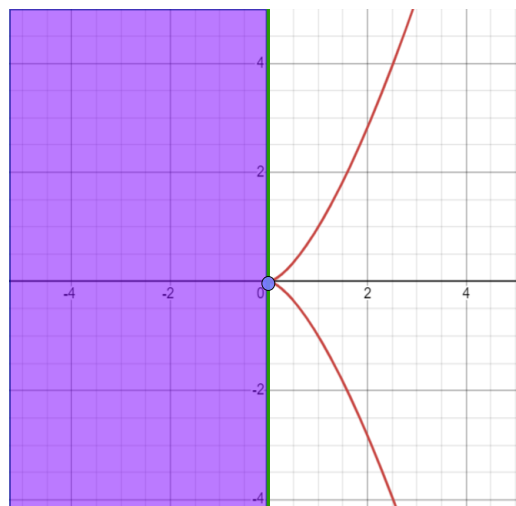


Figure 5.3: $x < 0$

Next, the root $x = 0$. This has 3 cells, the root itself and the two intervals on the y-axis split by the root $y < 0$, $y > 0$.

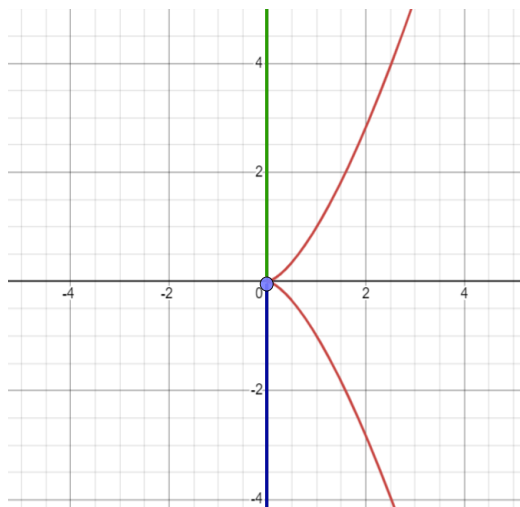


Figure 5.4: $x = 0$

The interval $x > 0$ has 5 cells. Two cells are the lines of the graph itself and the other three are the intervals between. Therefore, we have:

- Cell 1: $x > 0, y^2 - x^3 - x^2 > 0, y > 0$
- Cell 2: $x > 0, y^2 - x^3 - x^2 = 0, y > 0$
- Cell 3: $x > 0, y^2 - x^3 - x^2 < 0, y > 0$
- Cell 4: $x > 0, y^2 - x^3 - x^2 = 0, y < 0$
- Cell 4: $x > 0, y^2 - x^3 - x^2 > 0, y < 0$

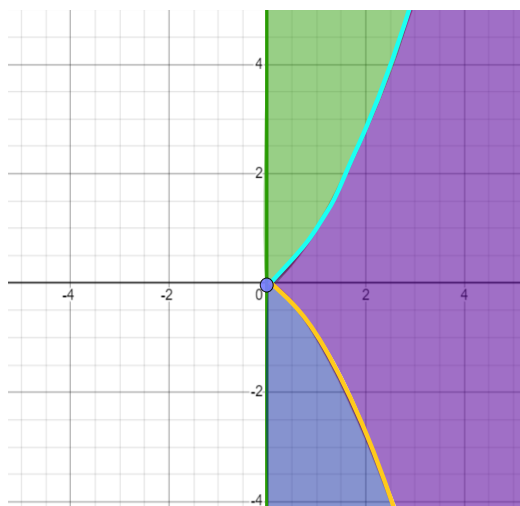


Figure 5.5: $x > 0$

Therefore, when $c = 0$ there are a total of 9 cells. Now, we can look at making c a small positive number, $c = 0.1$, altering the graph.

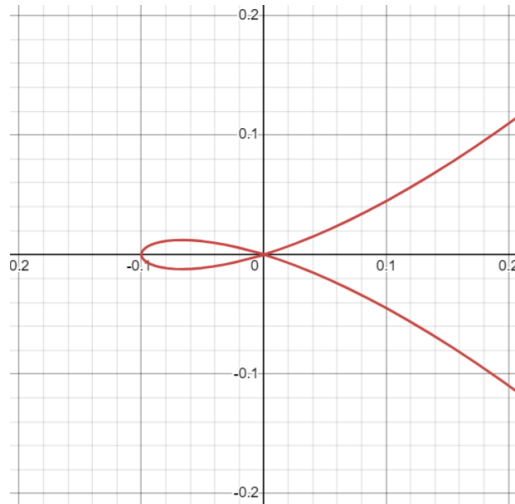


Figure 5.6: Graph of $ay^2 - bx^3 - cx^2$ when $a, b = 1, c = 0.1$

Here there are two roots, $x = -0.1$ and $x = 0$. Hence the proj to \mathbb{R}^1 consists of 5 cells $x < -0.1, x = -0.1, -0.1 < x < 0, x = 0, x > 0$.

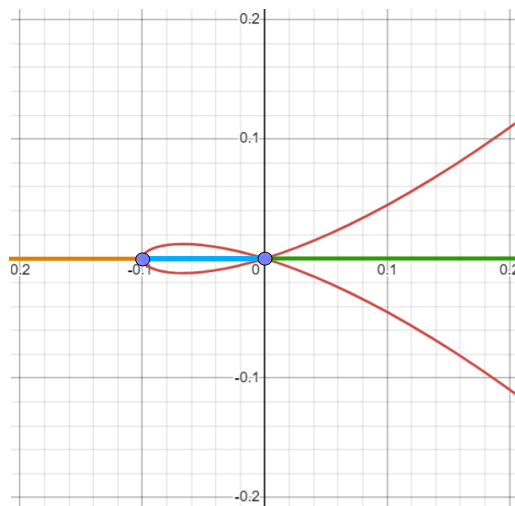


Figure 5.7: Projection onto the x-axis

We can expand this to \mathbb{R}^2 the same way. The interval $x < -0.1$ is simple as there is nothing set in the y-axis, therefore y is free (Figure 5.8(a)). The interval between the two roots $-0.1 < x < 0$ has 5 cells. Two cells are the lines of the graph itself and the other three are the intervals between (Figure 5.8(b)). The interval $x > 0$ has 5 cells

as well. Two cells are the lines of the graph itself and the other three are the intervals between (Figure 5.8(c)).

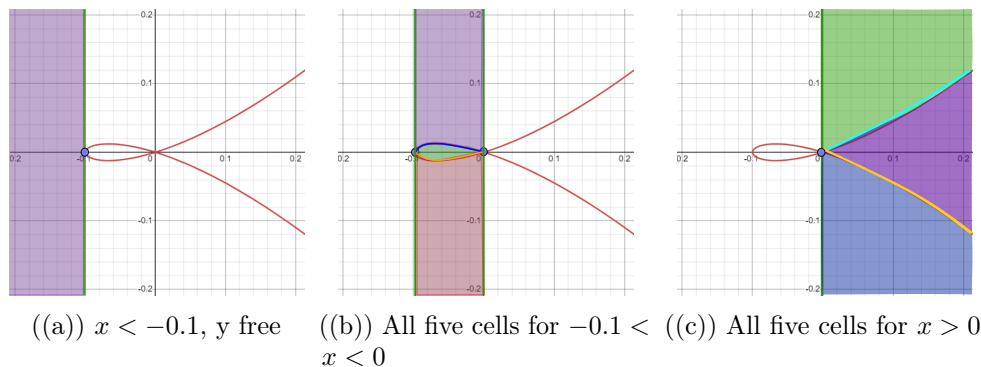


Figure 5.8: Cells over the intervals

Finally the two roots, $x = -0.1$ and $x = 0$. They both have 3 cells, the root itself and the two intervals on the y -axis split by the root $y < 0$, $y > 0$.

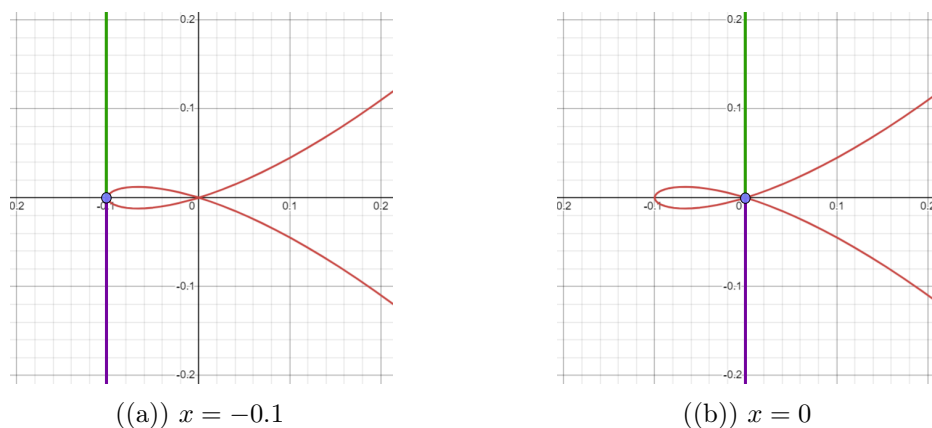


Figure 5.9: Cells over the roots

In total, there are 16 cells when $c > 0$.

Within this example, even if $c < 0$, there are fewer than 16 cells. The same goes for the values of a and b ; no matter what the three coefficients are, the maximum number of cells for this polynomial is 16. Our version of the CAD algorithm always outputs the 16 cells with 16 sample points. This does not affect the algorithm, as the sample points are always either from the root or within the intervals; therefore, we can end up with extra sample points. Take the graph in Figure 5.1: the interval $x < 0$ would have one sample point that is required from that interval. However, if we take all 16 cells from the graph in Figure 5.6, when $x < 0$, there would be three sample points: one for the root $x = -0.1$ and two from the intervals. Hence, if $c = 0$ and we take into account

the 16 cells, we end up with two extra sample points. Although there are extra sample points, they can still be used in the same way as the other interval sample points.

Corollary 5.23 From a finite tuple of polynomials, we can compute a finite tuple of polynomials including all their projections.

Proof. In order to check proj is computable we have to check $\text{proj}_1, \text{proj}_2, \text{proj}_3$ are computable using Lemma 5.19. \square

Definition 5.24 For a list of n -variate polynomials f_1, f_2, \dots, f_r , $x \in \mathbb{R}^n$. Let $\text{sign}(f_1, f_2, \dots, f_r, x) \in \{0, +, -\}^r$ be defined by $\text{sign}(f_1, f_2, \dots, f_r, x)(j) = +$ if $f_j(x) > 0$, $\text{sign}(f_1, f_2, \dots, f_r, x)(j) = -$ if $f_j(x) < 0$ and $\text{sign}(f_1, f_2, \dots, f_r, x)(j) = 0$ if $f_j(x) = 0$.

If a set of polynomials is delineable over a region \mathcal{R} , then the sign vector remains invariant over \mathcal{R} , [83, Lemma 1].

Lemma 5.25 Let $\mathcal{F} = \{f_1, \dots, f_r\} \subset \mathbb{R}[x_1, \dots, x_{n-1}][x_n]$ be a set of polynomials, and let $\text{proj}(\mathcal{F}) = \{q_1, \dots, q_r\} \subset \mathbb{R}[x_1, \dots, x_{n-1}]$ be the set of its projections. For any $b \in \{0, +, -\}^r$ we find that \mathcal{F} is delineable on $R_b := \{x \mid \text{sign}(q_1, \dots, q_r, x) = b\}$.

Proof. Following the structure of [81, Proof to Theorem 4.1] we show that the three properties (total number of complex roots, number of distinct complex roots, and number common complex roots) required to be invariant in Definition 5.20 can be expressed by referring to the signs of polynomials belonging to the projections.

1. Total number of complex roots of $f_i(y)$ remains invariant over \mathcal{R} .

That k_i is the number of roots of $f_i(y)$ is equivalent to the following:

$$(\forall k > k_i)[f_i^k(x_1, \dots, x_{n-1}) = 0] \wedge f_i^{k_i}(x_1, \dots, x_{n-1}) \neq 0$$

The formula depends on y only via the signs polynomials from $\text{proj}(\mathcal{F})$ take on it, which demonstrates the claim.

2. The total number of common complex roots of $f_i(y)$ and $f_j(y)$ remains invariant over \mathcal{R} .

The total number of common complex roots of two polynomials is the degree of their greatest common divisor. Knowing the psc of two polynomials allows us to know the degree of their gcd. For $f_i, f_j \in \mathcal{R}$ where $\text{deg}(f_i) = d_i$ and $\text{deg}(f_j) = d_j$, then for all $0 < m \leq \min(d_i, d_j)$

f_i and f_j have a common factor of degree equal to m

\iff

$\text{psc}_t(f_i, f_j) = 0, t = 0, \dots, m - 1$ and $\text{psc}_m(f_i, f_j) \neq 0$.

Therefore, $m_{i,j}$ being the number of common complex roots is expressed by the following:.

$$\begin{aligned} & (\forall k > k_i)[f_i^k(x_1, \dots, x_{n-1}) = 0] \wedge f_i^{k_i}(x_1, \dots, x_{n-1}) \neq 0 \wedge \\ & (\forall k > k_j)[f_j^k(x_1, \dots, x_{n-1}) = 0] \wedge f_j^{k_j}(x_1, \dots, x_{n-1}) \neq 0 \wedge \\ & (\forall m < m_{i,j})[\text{psc}_m^{x_n}(\hat{f}_i^{k_i}(x_1, \dots, x_n), \hat{f}_j^{k_j}(x_1, \dots, x_n)) = 0] \wedge \\ & [\text{psc}_{m_{i,j}}^{x_n}(\hat{f}_i^{k_i}(x_1, \dots, x_n), \hat{f}_j^{k_j}(x_1, \dots, x_n)) \neq 0] \end{aligned}$$

Again, we observe that the formula only depends on y via the signs polynomials from $\text{proj}(\mathcal{F})$.

3. The number of distinct complex roots of $f_i(y)$ remains invariant over \mathcal{R} .

We can express the number of distinct complex roots of a polynomial by looking at the number of common roots between it and its derivative. If k_i is the total number of complex roots of the polynomial and l_i is the number of common roots with its derivative, the number of distinct complex roots is $k_i - l_i$. Reusing ideas from above, this is expressed by:

$$\begin{aligned} & (\forall k > k_i)[f_i^k(x_1, \dots, x_{n-1}) = 0] \wedge f_i^{k_i}(x_1, \dots, x_{n-1}) \neq 0 \wedge \\ & (\forall l < l_i)[\text{psc}_l^{x_n}(\hat{f}_i^{k_i}(x_1, \dots, x_{n-1}), D_{x_n}(\hat{f}_i^{k_i}(x_1, \dots, x_n))) = 0] \wedge \\ & \text{psc}_{l_i}^{x_n}(\hat{f}_i^{k_i}(x_1, \dots, x_n), D_{x_n}(\hat{f}_i^{k_i}(x_1, \dots, x_n))) \neq 0 \end{aligned}$$

For a third time, we observe that the formula only depends on y via the signs polynomials from $\text{proj}(\mathcal{F})$.

□

Lemma 5.26 For each finite set \mathcal{F} of n -variate polynomials, there exists a sign invariant stack decomposition of \mathbb{R}^n .

Proof. The case $n = 1$ is immediate; we simply partition \mathbb{R} into the roots of the polynomials and the open intervals determined by them.

Otherwise, in the base phase of the CAD algorithm we repeatedly apply the projection operator $n - 1$ -times. We then obtain a decomposition of \mathbb{R}^1 which is sign-invariant for $\text{proj}^{n-1}(\mathcal{F})$.

We can extend a stack decomposition \mathcal{D}_{i-1} of \mathbb{R}^{i-1} which is sign invariant for $\text{proj}^{n-i+1}(\mathcal{F})$ to a stack decomposition \mathcal{D}_i of \mathbb{R}^i which is sign invariant for $\text{proj}^{n-i}(\mathcal{F})$: By Lemma 5.25 $\text{proj}^{n-i+1}(\mathcal{F})$ is delineable over each region of \mathcal{D}_{i-1} and hence the real roots of $\text{proj}^{n-i+1}(\mathcal{F})$ vary continuously over each region of \mathcal{D}_{i-1} , while maintaining their order (cf. [107, Corollary 8.6.5]). □

Definition 5.27 A representative sample for a list of n -variate polynomials is a finite set of points X , such that for every region \mathcal{R} of the sign invariant stack decomposition provided by Lemma 5.26, with $\mathcal{R} \cap [0, 1]^n \neq \emptyset$ there exists $x \in \mathcal{R} \cap X$.

Lemma 5.28 Let X be a representative sample for $\text{proj}(f_1, f_2, \dots, f_r)$. Then there is a representative sample X' for f_1, f_2, \dots, f_r such that

$X = \{(x_1, \dots, x_{n-1}) \mid \exists x_n. (x_1, \dots, x_{n-1}, x_n) \in X'\}$. Moreover, X' can be obtained as follows: For each $\bar{x} \in X$, let $S_{\bar{x}}$ be a representative sample for the univariate polynomials $f_1(\bar{x}), \dots, f_r(\bar{x})$, and then let $X' = \{(\bar{x}, x_n) \mid \bar{x} \in X \wedge x_n \in S_{\bar{x}}\}$.

Proof. In the representative sample $X = \{a_1, \dots, a_q\}$, each a_i is a set of points for each region of the projections, therefore a root or a point within the interval between two non-trivial roots. We select a set of test points $b = (b_1, \dots, b_p)$ for each region of the original polynomials.

We will construct the sample points of the regions of \mathcal{D}_i which belong to the stack over the region $\mathcal{C} \subset \mathcal{D}_{i-1}$. The polynomials in $\text{proj}^{n-1}(\mathcal{F})$ can be evaluated over the sample point, resulting in a set of univariate polynomials in x_i . These univariate polynomials can now be treated the same as the projections, where we can isolate the roots and a representative sample.

In order to extend this to \mathbb{R}^n , we can substitute a_i into our original polynomial in order to achieve a set of r univariate polynomials in x_n , which we will denote $F_a = \{f_1(a_i), \dots, f_r(a_i)\}$. We can also substitute in $\text{proj}(b)$ for a second set of r univariate polynomials, which we will denote $F_b = \{f_1(\text{proj}(b)), \dots, f_r(\text{proj}(b))\}$. This will allow us to compare two polynomials, one from each set F_a, F_b , in order to find a suitable point of x_n which will result in the same sign vector. As our test points b are already in \mathbb{R}^n , we already know $\text{sign}(f_1, \dots, f_r, b)$ and hence know what sign we need the univariate polynomials, F_a to be for the sign vectors to be equal.

By Lemma 5.25, the choice of the points a_i makes sure the univariate polynomials F_a have the same total number of complex roots, and the same number of distinct complex roots as our original polynomials ([81]'s proof to Theorem 4.1 excludes the zero polynomial, however, the argument still works). If the total number of complex roots is odd, the polynomial has at least one real root with an odd multiplicity. If the total number of complex roots is even, then there could be no real roots or real roots with even multiplicity.

For an F_b with all three states (positive, negative, and zero sign vector), we would need to confirm our F_a can also take all three states. This can be achieved by checking the multiplicity of the roots. The polynomial has a root with an odd multiplicity iff it crosses the axis. We know what sign the original polynomial will give from $\text{sign}(f_1, \dots, f_r, b)$ allowing us to find the condition needed on the point x_n to give F_a in order for $\text{sign}(f_1, \dots, f_r, (a_i, x_n)) = \text{sign}(f_1, \dots, f_r, b)$.

If it is the zero polynomial, we can select any point as our x_n and the sign vectors would be equal.

If the univariate polynomials have an even multiplicity they would either have a sign vector $\{0, +\}$ or $\{0, -\}$. We would confirm what sign this need to be using $\text{sign}(f_1, \dots, f_r, b)$, which will allow us to find the condition on the point x_n . A similar occurrence happens when the univariate polynomials are always positive or always negative, as $\text{sign}(f_1, \dots, f_r, b)$ confirms what sign we require. \square

Theorem 5.29 There is a computable procedure that takes as input a finite list of n -variate real polynomials and outputs a finite list (I_0, \dots, I_ℓ) of $\text{AoUC}_{[0,1]^n}$ -instances such that

$$\{x \in [0, 1]^n \mid \exists i I_i = \{x\}\}$$

is a representative sample for the polynomials.

Proof. The base case is trivial. The unique point $0 \in \mathbb{R}^0$ forms a representative sample for any collection of zero-variate polynomials. We can just output an $\text{AoUC}_{[0,1]}$ -instance $\{0\} \in \mathcal{A}(\mathbb{R}^0)$.

Given a finite list of $n + 1$ -variate real polynomials, we can compute a finite list of n -variate polynomials including their projections using Corollary 5.23. By the induction hypothesis, we can compute finitely many $\text{AoUC}_{[0,1]^n}$ -instances such that the determined outputs form a representative sample for the projections.

By considering the upper bounds on the ranks available to us, we can obtain some upper bound on the number of $\text{AoUC}_{[0,1]^{n+1}}$ -instances required in advance. □

Corollary 5.30 $\text{AoUC}_{[0,1]}^* \leq_W \text{BMRoot} \leq_W \text{BPIneq} \leq_W \text{AoUC}_{[0,1]}^\diamond$.

Proof. The first reduction is a consequence of Proposition 5.12.

Asking for a root of a polynomial P is the same as asking for a solution of $P(\bar{x}) \geq 0 \wedge -P(\bar{x}) \geq 0$; this shows the second reduction.

For the third, we observe that if there is any solution to $\bigwedge_{i \leq k} P_i(\bar{x}) \geq 0$ within $[0, 1]^n$, then every representative sample for P_0, P_1, \dots, P_k contains a solution. By Theorem 5.29, $\text{AoUC}_{[0,1]}^*$ lets us obtain a representative sample. Non-solutions will eventually be recognized as such, which is why $\bigsqcup_{n \in \mathbb{N}} C_n$ can identify a correct solution from finitely many candidates. As shown in [86], it holds that $(\bigsqcup_{n \in \mathbb{N}} C_n) \star \text{AoUC}_{[0,1]}^* \equiv_W \text{AoUC}_{[0,1]}^\diamond$. □

We are now prepared to prove our upper bound for the Weihrauch degree of finding Nash equilibria in multiplayer games:

Corollary 5.31 $\text{Nash} \leq_W \text{AoUC}_{[0,1]}^\diamond$.

Proof. A strategy profile is a set of strategies for all players which fully specify all actions in a game. We say that a strategy profile is supported on a set S of actions if every action outside of S has probability 0 in the strategy profile, and for every player the expected payoff for actions in S is at least as much as for every other action. A strategy profile is a Nash equilibrium if and only if it is supported on some set S .

For a fixed set S (for which there are only finitely many candidates), the property of being a strategy profile on it can be expressed as a multivariate polynomial system of inequalities. By compactness, we can detect if such a system has no solution in $[0, 1]^n$. This means that $(\bigsqcup_{n \in \mathbb{N}} C_n)$ can be used to select a set S with the property that some strategy profile is supported on it. We know from Nash's theorem that such a set S must exist.

Once we have selected a suitable \mathcal{S} , we invoke BPIneq to actually get a strategy profile supported on it. This is a Nash equilibrium, as desired. We thus get (taking into account Corollary 5.30):

$$\text{Nash} \leq_{\text{W}} \text{BPIneq} \star \left(\bigsqcup_{n \in \mathbb{N}} C_n \right) \leq_{\text{W}} \text{AoUC}_{[0,1]}^{\diamond}$$

□

5.3.4 Differences Between our Algorithm and the Original

We define sections and sectors in Definition 5.13 which are sets of continuous real-valued function on the region \mathcal{R} . Therefore, the stack decomposition which consists of sections and sectors (Definition 5.14) is also built around functions. We clarify that we speak of an *algebraic* stack decomposition, if these continuous functions are actually all polynomials.

In the original CAD algorithm the decomposition (Definition 5.14) has disjoint regions. Our algorithm, however, can get multiple copies of the same regions.

Definition 5.32 Let $\mathcal{R} \subseteq \mathbb{R}^n$. A *weak* decomposition of \mathcal{R} is a finite collection of regions $(R_i)_{i \in I}$ whose union is \mathcal{R} subject to $\mathcal{R}_i \cap \mathcal{R}_j = \emptyset$ or $\mathcal{R}_i = \mathcal{R}_j$ for all $i, j \in I$.

A weak algebraic stack decomposition of \mathbb{R}^0 is just a weak decomposition of \mathbb{R}^0 . A weak algebraic stack decomposition of \mathbb{R}^{n+1} is a weak decomposition of the form $\bigcup_{\mathcal{R} \in \mathcal{D}} S_{\mathcal{R}}$, where each $S_{\mathcal{R}}$ is an algebraic stack over \mathcal{R} and \mathcal{D} is a weak algebraic stack decomposition of \mathbb{R}^n .

Definition 5.33 A (weak) algebraic stack decomposition is minimal for a given set of polynomials \mathcal{F} if each $p \in \mathcal{F}$ is delineable on it, but removing any polynomial from the (weak) algebraic stack decomposition breaks this property.

If we could test for equality, CAD could produce a minimal algebraic stack decomposition. However, we cannot do this as we do not know the number of pieces/degrees/roots of the polynomials.

Example 5.34 Consider $\mathcal{F} = \{x^2 + \epsilon\}$ for some parameter $\epsilon \in \mathbb{R}$. If ϵ is positive, a minimal algebraic stack decomposition for \mathcal{F} uses 0 polynomials, if $\epsilon = 0$ we need 1, and if ϵ is negative we need 2. Therefore, LPO reduces to finding minimal algebraic stack decompositions already in the simplest case.

An algebraic stack decomposition (not weak) for \mathcal{F} consists of a finite tuple of real numbers (x_1, \dots, x_k) that are promised to be distinct and contain the roots of $x^2 + \epsilon$. This allows us to check if ϵ is zero or negative: if $\epsilon < \frac{1}{2} \min_{i,j \leq k, i \neq j} |x_i - x_j|^2$, it already has to be the case that $\epsilon = 0$. So even just asking for a algebraic stack decomposition of a single univariate polynomial requires solving LPO.

Theorem 5.35 Given a finite set of multivariate polynomials, we can compute a weak algebraic stack decomposition such that the original polynomials are delineable over the stack.

Proof. The basic idea of the CAD algorithm is that for a finite collection \mathcal{F} of n -variate polynomials, we obtain a weak algebraic stack decomposition as $\text{proj}^n \mathcal{F}, \text{proj}^{n-1} \mathcal{F}, \dots, \text{proj} \mathcal{F}$. Any finite overapproximation of the projections still works, thus Lemma 5.19 yields the claim. \square

5.4 An Open Question and a Remark

Our main theorem demonstrates that $\text{AoUC}_{[0,1]}^* \leq_w \text{Nash} \leq_w \text{AoUC}_{[0,1]}^\diamond$ which immediately raises the question of which of those reductions are strict. As previously mentioned, according to [86] it is established that $\text{AoUC}_{[0,1]}^* <_w \text{AoUC}_{[0,1]}^\diamond$, implying that at least one of the two reductions is strict. Since the degrees of $\text{AoUC}_{[0,1]}^*$ and $\text{AoUC}_{[0,1]}^\diamond$ exhibit significant similarities in various aspects, only a few of the established techniques are available to resolve this situation. While the use of the recursion theorem as demonstrated in [86, 88] might be possible, it certainly presents a considerable challenge.

A similar question was left unresolved in [86, Section 5]. In that work, two variants of Gaussian elimination were defined as follows:

Definition 5.36 ([86]) $\text{LU-Decomp}_{P,Q}$ takes as input a matrix A , and outputs permutation matrices P, Q , a matrix U in upper echelon form and a matrix L in lower echelon form with all diagonal elements being 1 such that $PAQ = LU$. By LU-Decomp_Q we denote the extension where P is required to be the identity matrix.

While $\text{LU-Decomp}_{P,Q} \equiv_w \text{AoUC}_{[0,1]}^*$ was demonstrated, for the other variant only $\text{AoUC}_{[0,1]}^* \leq_w \text{LU-Decomp}_Q \leq_w \text{AoUC}_{[0,1]}^\diamond$ could be established. A clearer understanding of situations where sequential uses of $\text{AoUC}_{[0,1]}$ are genuinely required to perform some “algorithm”, and ideally a mathematical theorem or simpler problem which is equivalent to $\text{AoUC}_{[0,1]}^\diamond$ both seem to be very desirable.

Should it hold that $\text{AoUC}_{[0,1]}^* <_w \text{Nash}$, it would be very interesting to see how many players are needed to render the Weihrauch degree of finding Nash equilibria harder than the two-player case. A natural conjecture would be that this already occurs for three players. An important distinction between two-player and three-player games is that two-player games with rational payoffs have rational Nash equilibria, while for every algebraic number $\alpha \in [0, 1]$ there is a three-player game where every Nash equilibrium assigns α as a weight to a particular action, as shown by Bubelis [35]. However, the construction employed by Bubelis does not yield a reduction $\text{BRoot} \leq_w \text{Nash}_3$, as it requires a polynomial with α as a simple root as starting point. On the other hand, we know that even $\text{BRoot} \leq_w \text{Nash}_2$ holds via Corollary 5.8, albeit with a very roundabout construction. Thus, this particular difference between two and three-player games is immaterial to the Weihrauch degrees concerned.

5.4.1 Route not Taken: ‘Potential’ Univariate Polynomials

In the intermediate stages of this research, we explored the CAD procedure from a different angle to which we achieved our end results. However, the work completed within this tangent is in itself an interesting subject and is laid out within this section.

Within the CAD algorithm, we can encounter the following scenario: We have an $\text{AoUC}_{[0,1]^n}$ -instance and a $n + 1$ -variate polynomial. If the former indeed specifies a unique point, then we want to substitute that point for the first n variables in the polynomial and compute the roots of the resulting univariate polynomial. The beauty lies in the fact that the interplay of addition, multiplication, and the $\text{AoUC}_{[0,1]}$ principle align harmoniously, enabling us to postpone any non-computable operations until the very last phase. The subsequent portion of this subsection will meticulously expound upon this concept.

Definition 5.37 A *potential univariate polynomial* is a vector of $\text{AoUC}_{[0,1]}$ -instances $I - i$, along with a $k \in \mathbb{N}$. If each I_i collapses to some $\{a_i\}$, the potential polynomial specifies the “actual” polynomial $\sum_{i \leq n} (2ka_i - k)x^i$.

The potential polynomials, being an $\text{AoUC}_{[0,1]}$ -instance, they are restricted between $[0, 1]$, while the ‘actual’ polynomials are not. Therefore, to solve this issue we specify them using $\sum_{i \leq n} (2ka_i - k)x^i$. We can expand the operations of addition and multiplication for polynomials to encompass potential polynomials, all the while preserving the integrity of the specification relation.

Lemma 5.38 There are computable operations \oplus and \star defined on potential univariate polynomials such that if the potential polynomial P_i specifies the polynomial p_i , then $P_0 \oplus P_1$ specifies $p_0 + p_1$ and $P_0 \star P_1$ specifies $p_0 \times p_1$.

Proof. We are given two potential polynomials P_0 and P_1 , where P_0 has the coefficients bound between $[-k, k]$, while those of P_1 are bounded between $[-j, j]$. We use these as input, with the output for addition being the full interval $[-k - j, k + j]$.

Our output is the bound plus $\text{AoUC}_{[0,1]}$ -instances. The k^{th} $\text{AoUC}_{[0,1]}$ -instances are constructed by looking at the two k^{th} $\text{AoUC}_{[0,1]}$ -instances from the input potential polynomials. If both of these instances do not specify a point, we do nothing. However, if they both specify a point, the k^{th} $\text{AoUC}_{[0,1]}$ -instance we build for the output should also specify a point within the bounds.

If we have two points a, b where $a \in [0, 1]$ and $b \in [0, 1]$ which can be scaled up to the true bounds of P_0 and P_1 , giving $(2a - 1)k$ and $(2b - 1)j$. These bounds can be added together, $2(ka + jb) - (k + j)$ to show the true coordinate of the point. In order to calculate the non-scaled point within $[0, 1]$, we introduce x which scales to $2x(k + j) - (k + j)$. Therefore, $2(ka + jb) - (k + j) = 2x(k + j) - (k + j)$, hence $x = \frac{ka + jb}{k + j}$.

Multiplication is influenced by all of the coefficients of the potential polynomials. If we know the degrees of the potential polynomials and we can specify every coefficient from the input P_0 and P_1 , then we can compute the bounds for the output. This is achieved by multiplying the coefficients of P_0 and P_1 together and then summing them.

Therefore, the bound would be Mkj where $M = \max(\deg P_0, P_1)$.

Similarly to the addition, we can take points $a_0, \dots, a_m, b_0, \dots, b_m \in [0, 1]$ from P_0, P_1 which relate to the coefficients of each polynomial with m being the bound of the degree of the polynomials. These can be scaled up to the bounds of P_0, P_1 giving $(2a_m - 1)k$ and $(2b_m - 1)j$. These bounds can be multiplied together to show the true coordinate of the point, $(2a_m - 1)(2b_m - 1)kj$. As a way of calculating the non-scaled point within $[0, 1]$, we can introduce x which scales to $(2x - 1)Mkj$.

Calculating the non-scaled points within $[0, 1]$ becomes more complex with multiplication, as outputs may need multiple inputs to be summed together. The summation we are looking at is $\sum_{i=0}^m a_i b_{m-i}$ which, once scaled up, becomes $\sum_{i=0}^m (2a_i - 1)(2b_{m-i})kj$. We can rearrange this in order to get the unscaled and mixed construction, and make it equal to our scaled x :

$$4kj \left(\sum_{i=0}^m a_i b_{m-i} \right) - 2kj \left(\sum_{i=0}^m a_i + b_{m-i} \right) + mkj = 2Mkjx - Mkj$$

$$x = \frac{2}{M} \left(\sum_{i=0}^m a_i b_{m-i} \right) - \frac{1}{M} \left(\sum_{i=0}^m a_i + b_{m-i} \right) + \frac{M + m}{2M}$$

□

Since we do not require these operations to work in a precise way for non-collapsing $\text{AoUC}_{[0,1]}$ -instances, we do not need to employ interval arithmetic here. We can also extend root-finding, in the following way:

Lemma 5.39 Given a finite list of potential univariate polynomials, $\mathcal{F} = \{f_1, f_2, \dots, f_r\}$, we can use $\text{AoUC}_{[0,1]}^*$ to compute a finite list of points, $X = \{x_1, x_2, \dots, x_l\}$ such that every root of a determined and non-zero polynomial in \mathcal{F} is in the list.

Proof. Similarly to the proof of Proposition 5.5. We have an upper bound on the degree of each potential polynomial available, and allocate 3^k $\text{AoUC}_{[0,1]}$ -instances to a polynomial of degree at most k . We monitor each potential polynomial, until we have found confirmation that it is an actual polynomial, which furthermore is not constant 0. If this never happens, the corresponding $\text{AoUC}_{[0,1]}$ -instances remain unspecified.

If we know that we have an actual non-constant 0 polynomial p , we identify some $a \leq 0$ and $b \geq 1$ such that $p(a) \neq 0 \neq p(b)$. We can then invoke the algorithm from [96, Theorem 4.1] and compute a list of 3^k numbers containing all zeros of the polynomial. We specify the corresponding $\text{AoUC}_{[0,1]}$ -instances to take the corresponding values, defaulting to 0 or 1 if the value falls outside of $[0, 1]$. □

5.5 Consequences of the Classification

In this section, we shall explore some consequences of our classification of the Weihrauch degree of finding Nash equilibria. For this, we consider more permissive notions of

algorithms and whether or not they are sufficiently powerful to solve the task. The first important point, however, is that the non-computability of finding Nash equilibria is inherently tied to the potential of having multiple Nash equilibria.

Corollary 5.40 Let $f : \mathbf{X} \rightarrow \mathbf{Y}$ be a function where \mathbf{Y} is computably admissible. Then if $f \leq_W \text{Nash}$, then f is already computable.

Proof. By combining Theorem 5.3 with [96, Theorem 2.1] (originally [20, Theorem 5.1]), since $\text{AoUC}_{[0,1]}^\diamond \leq_W C_{\{0,1\}^{\mathbb{N}}}$. \square

An immediate consequence of Corollary 5.40 is that if we restrict our consideration to games having a unique Nash equilibrium, then computing the Nash equilibrium is possible. However, this insight goes even further. For example, we could consider the class of games where Player 1 receives the same payoff in any Nash equilibrium. Then computing the equilibrium payoff for Player 1 is possible, even if we might be unable to compute a Nash equilibrium.

As our first extended notion of algorithm, we consider computation with finitely many mind changes. We begin with a model of computation where the machine continues to output more and more digits of the infinite code for the desired output. We then add the ability for the machine to completely erase all digits written so far, and to start over. To ensure that there is a well-defined output, this ability may be invoked only finitely many times. It was shown in [20, 28] that a problem f is solvable with finitely many mind changes iff $f \leq_W C_{\mathbb{N}}$.

Corollary 5.41 Nash is solvable with finitely many mind changes.

We can delve a bit deeper and obtain an upper bound for the number of mind changes required from the dimensions of the game.

Next, we consider various probabilistic models of computation. A Las Vegas machine can use random coin flips to help with its computation. At any point during the computation, it can report a fault and abort, but if it continues running forever, it needs to produce a valid output. For each input, the probability (based on the coin flips) of outputting a correct output needs to be positive (but we do not demand a global positive lower bound). This model was introduced in [23]. Since Las Vegas computability is closed under composition, we obtain the following strengthening to their [23, Corollary 17.3] (by using their [23, Corollary 16.4]):

Corollary 5.42 Nash is Las Vegas computable.

It was previously demonstrated in [23, Theorem 16.6] that even for a Las Vegas computation solving just $\text{AoUC}_{[0,1]}$ it is not possible to compute a positive lower bound for the success chance from the input – so in particular, there is no global lower bound.

By dropping the requirement that a wrong guess must be reported at some stage of the computation, we arrive at Monte Carlo machines. They, too, make random coin tosses and are subject to the requirement that any completed output must be correct and that a correct output needs to be given with some positive probability. However,

they can fail by simply stopping to produce output, due to the Halting problem, we cannot detect whether they have done that. This model was introduced in [26], and from the characterizations obtained there together with our classification it follows that:

Corollary 5.43 Nash is Monte Carlo computable, and moreover, we can compute a positive lower bound for the success chance from the dimensions of the game.

Indeed, in the context of probabilistic computation for finding Nash equilibria, we are confronted with a choice. We can opt for either possessing knowledge of a lower bound on the success probability or being able to detect when a random guess during the computation proves to be incorrect. This choice highlights a trade-off between the two aspects within the confines of probabilistic algorithms aimed at solving this problem.

Part IV

Machine Learning and Cybersecurity Applications

Chapter 6

Introduction

Machine learning and cybersecurity have emerged as dynamic and rapidly evolving research domains, poised to shape the technological landscape of the future. Machine learning techniques have entered various industries in order to establish systems to learn from data and improve their performance over time. In the context of cybersecurity, machine learning plays a crucial role in identifying and mitigating threats [3], detecting anomalies [111], and adapting to new attack patterns [68]. The vast amounts of data generated in the digital era have propelled machine learning to new heights, enabling the development of sophisticated algorithms for enhancing security measures. Digital data has increased from 2 Zettabytes in 2010 and is predicted to be 175 Zettabytes of data by 2025 [76], with 87 Zettabytes residing in the public cloud and 79.4 Zettabytes created by IoT devices.

Furthermore, cybersecurity has become a critical concern as society becomes increasingly dependent on digital platforms. With the rise of cyber attacks, data breaches, and online vulnerabilities, there is a pressing need to develop advanced solutions to safeguard sensitive information and digital infrastructures. Microsoft Digital Defense Report 2022 [106] shows how much the nation state groups' targeting of critical infrastructure increased in the past year with actors' focusing on companies in the IT sector, financial services, transportation systems and communications infrastructure. As well, there is a 900% year over year increase in proliferation of deepfakes since 2019. This has led to a surge in research efforts focused on enhancing cybersecurity mechanisms to stay ahead of evolving threats. However, the UK Government's Cyber Security Breaches Survey 2022 [103] reported that the number of cyber attacks remained consistent with 2021. They noted that less cyber-mature organisations may be under-reporting, as organisations with enhanced cybersecurity usually result in a higher identification of attacks. This also shows how the need for education of cyber attacks for digital platforms is important.

In this landscape, computable analysis and game theory offer fresh perspectives that can contribute to both machine learning and cybersecurity. Computable analysis provides a rigorous framework for analysing the computational complexity of algorithms and their convergence properties. In the context of machine learning, this can help in understanding the existence, efficiency and reliability of learning algorithms, leading

to more effective and trustworthy models. Additionally, game theory's application in cybersecurity can assist in modelling adversarial scenarios, predicting potential attack strategies, and designing robust defence mechanisms. By considering interactions between attackers and defenders as strategic games, researchers can devise strategies that minimise risks and losses in cyber operations.

In conclusion, the intersection of machine learning, cybersecurity, computable analysis, and game theory presents a rich avenue for innovative research. The insights gained from these disciplines can revolutionise the ways we address security challenges, ensuring the safety and reliability of digital systems in an increasingly interconnected world.

Chapter 7

A Computable Analysis Perspective on Verified Machine Learning

Contents

7.1	Introduction	79
7.2	A Theory of Verified Machine Learning	80
7.3	Related Work	93
7.4	Conclusions	94

This chapter (with the exception of Section 7.3.6) is based on the paper ‘A Computability Perspective on (Verified) Machine Learning’ published in WADT [47]. It is joint work with Jay Morgan, Arno Pauly, and Markus Roggenbach.

7.1 Introduction

Machine Learning (ML) involves the creation of predictive and generative models using optimisation techniques. The remarkable achievements of ML methods across diverse domains prompt the question of how much confidence one can place in the outcomes produced by an ML model. Given the application of ML models in critical domains, some level of verification becomes essential, as eloquently argued by Kwiatkowska [94].

However, the extensive reliance on non-discrete mathematics in ML renders traditional verification techniques challenging to apply to its artifacts. Moreover, numerous ML applications lack specifications in forms such as an input/output relationship, which often serve as foundations for ‘classical’ verification methods. For instance, consider an ML application designed to determine whether a given image depicts a cat. In the absence of a clear specification, the question arises: what type of properties can be effectively verified? In our perspective, akin to classical verification, broadening the scope of verifiable properties beyond simple input/output relationships can prove valuable.

By leveraging the tools of computable analysis, which deals with computation on continuous data types, we embrace the same realm of continuous mathematics that underpins the theory of machine learning. This approach not only sidesteps ad-hoc discretisation but also aligns seamlessly with the continuous nature of ML’s mathematical foundations.

We embark on an exploration into the realm of verifiable questions concerning ML models, a pursuit independent of the specific ML framework employed. These questions, we believe, serve as fundamental components for shaping a future property specification language dedicated to ML. The task of discretisation, while possibly necessary for efficiency, can be delegated to the implementation phase, all while preserving correctness.

To formally characterise the computational queries we aim to address, we employ the language of computable analysis. We establish their general solvability through algorithmic demonstrations, ensuring their applicability across diverse ML methodologies. It’s important to note that the semi-decision procedures presented are not designed for direct implementation purposes. Furthermore, we refrain from making assertions about computational complexity.

7.2 A Theory of Verified Machine Learning

One fundamental concept in ML is that of a classifier. A classifier takes as input a description of an object, often in the form of a vector of real numbers, and provides an output that is either a colour or no response. This notion positions classifiers as a generalisation of semi-decision procedures. Within the context of computable analysis, the theory centres around functions defined on real numbers and other sets derived from analysis, which can be computed by machines. We use the theory from Chapter 2 as base knowledge for this chapter, utilising represented spaces.

7.2.1 A Theory of Classifiers

We focus exclusively on classification tasks. In this context, a trained model takes a description of an object as input and produces an output that is either a class (usually represented as an integer from $\mathbf{k} = 0, \dots, k - 1$, where $k > 0$) or provides no response. It’s important to note that the absence of a response might arise due to the algorithm’s failure to terminate, rather than a deliberate refusal to select a class. This consideration is significant when dealing with domains like the real numbers, given the continuity of all computable functions. To formally handle this, we work with the represented space \mathbf{k}_\perp , encompassing elements such as $0, \dots, k - 1, \perp$, where 0^ω exclusively represents \perp , and any $0^m 1^\ell 0^\omega$ stands for $\ell < k$, with $m \in \mathbb{N}$.

In the realm of implementable classifiers, they must be computable functions. An essential observation from computable analysis is that computable functions must inherently be continuous. Interestingly, the most suitable concept of a function space within computable analysis is the space $\mathcal{C}(\mathbf{X}, \mathbf{Y})$, housing continuous functions from \mathbf{X} to \mathbf{Y} .

Definition 7.1 A *classifier* is a (computable/continuous) procedure that takes some $x \in \mathbf{X}$ as input, and either outputs a colour $j \in \mathbf{k}$, or diverges (which is seen as outputting \perp). The collection of classifiers is the space $\mathcal{C}(\mathbf{X}, \mathbf{k}_\perp)$.

Any concrete classifier we would care about will actually be computable. However, the definition includes also non-computable (but continuous) ones.

Example 7.2 Consider the classifier we would obtain from Support Vector Machine (SVM) [71]. The relevant space \mathbf{X} will be \mathbb{R}^n for some $n \in \mathbb{N}$. The classifier is described by a hyperplane P splitting \mathbb{R}^n into two connected components C_0 and C_1 . We have two colours, so the classifier is a map $p : \mathbb{R}^n \rightarrow \mathbf{2}_\perp$. If $x \in C_i$, then $p(x) = i$. If $x \in P$, then $p(x) = \perp$.

On the fundamental level, we need the *no-answer answer* \perp as we will never be able to be certain that a numerical input is exactly on the separating hyperplane, even if we keep increasing the precision: equality on reals is not decidable.

Practically, computations might be performed using floating-point arithmetic, where equality is decidable. In this, the use of \perp is still meaningful: If we keep track of the rounding errors encountered, we can use \perp to denote that the errors have become too large to classify an input.

Example 7.3 Neural network classifiers compute a class score for every colour, which, when these class scores are normalised, share similar properties as a probability distribution. This translates into our framework by fixing a threshold $p \geq 0.5$, and then assigning a particular colour to an input iff its class score exceeds the threshold p . If no colour has a sufficiently high score, the output is \perp . As long as the function computing the class scores is computable, so is the classifier we obtain in this fashion. If our class scores can use arbitrary real numbers, we cannot assign a colour for the inputs leading to the exact threshold.

When considering a classifier, the range of verification questions we can pose is extensive. These questions aim to validate specific criteria or assess the classifier's performance over a designated set or region A . Verification serves the purpose of ensuring system correctness and identifying potential flaws. Here are concrete examples of the questions we might ask:

Proposition 7.4. The following maps are computable:

1. $\text{existsValue} : \mathbf{k} \times \mathcal{V}(\mathbf{X}) \times \mathcal{C}(\mathbf{X}, \mathbf{k}_\perp) \rightarrow \mathbb{S}$, which answers true on input (n, A, f) iff $\exists x \in A f(x) = n$.
2. $\text{forallValue} : \mathbf{k} \times \mathcal{K}(\mathbf{X}) \times \mathcal{C}(\mathbf{X}, \mathbf{k}_\perp) \rightarrow \mathbb{S}$, which answers true on input (n, A, f) iff $\forall x \in A f(x) = n$.
3. $\text{fixedValue} : \mathbf{k} \times (\mathcal{V} \wedge \mathcal{K})(\mathbf{X}) \times \mathcal{C}(\mathbf{X}, \mathbf{k}_\perp) \rightarrow \mathbf{2}_\perp$, which on input (n, A, f) answers 1 iff $\forall x \in A f(x) = n$, and answer 0 iff $\exists x \in A f(x) \in \mathbf{k} \setminus \{n\}$, and \perp otherwise.

4. $\text{constantValue} : (\mathcal{V} \wedge \mathcal{K})(\mathbf{X}) \times \mathcal{C}(\mathbf{X}, \mathbf{k}_\perp) \rightarrow \mathbf{2}_\perp$, which on input (A, f) answers 1 iff there is some $n \in \mathbf{k}$ such that $\text{fixedValue}(n, A, f)$ answers 1, and which answers 0 iff $\text{fixedValue}(n, A, f)$ answers 0 for all $n \in \mathbf{k}$.

Proof. 1. Given a colour $k \in \mathbf{k}$ and a classifier $f \in \mathcal{C}(\mathbf{X}, \mathbf{k}_\perp)$, we can compute $f^{-1}(k) \in \mathcal{O}(\mathbf{X})$ due to the discreteness of \mathbf{k} . By definition of overtness, given $A \in \mathcal{V}(\mathbf{X})$, we can then semidecide whether $A \cap f^{-1}(k) \neq \emptyset$.

2. As above, we can compute $f^{-1}(k) \in \mathcal{O}(\mathbf{X})$. By the definition of compactness, given $A \in \mathcal{K}(\mathbf{X})$ we can semidecide whether $A \subseteq f^{-1}(k)$.
3. By running the algorithms from 1. and 2. in parallel.
4. Since \mathbf{k} is both compact and overt, we can quantify over it both existentially and universally. Or, more down to earth, we can simply run the algorithm from 3. for all finitely many cases in parallel, and answer once we have received enough information from those runs.

□

Here we can present real-world examples of situations where these questions could prove useful:

1. existsValue :

Example 7.5 Consider a DDOS-attack detection system realised as a classifier f . The region A consists of data indicating an attack is happening, and the colour n means that the system concludes there is no attack. If $\text{existsValue}(n, A, f)$ returns true, we have identified a false negative in f .

2. forallValue :

Example 7.6 Continuing with example 7.5, here we may consider the property that for every data point in the region A the presence of the attack is successfully identified. If $\text{forallValue}(n, A, f)$ returns true, we have verified that all points in the set A are classified as expected.

3. fixedValue :

Example 7.7 Consider an automated stock trading system which makes decisions to buy, sell or hold a particular stock based on its technical indicators. Deciding to buy or sell is represented as a colour (as it is an active decision), while \perp means to hold the current position. Given a region A of very positive technical indicator values, we may want to ideally be assured that the system will always buy, while the decision to sell would be a clear mistake. If $\text{fixedValue}(\text{buy}, A, f)$ returns 1, we know that the system meets the ideal requirement. If it returns 0, we have found a mistake. Answer \perp means that the system falls short of its target without making a clear mistake.

4. `constantValue`:

Example 7.8 Assume we have reason to believe that all points in the region A are very similar, and should thus be classified in the same way by the classifier f . If `constantValue(A, f)` returns 1, we have confirmation that this indeed happens. Obtaining the answer 0 suggests that a mistake might have happened, as two similar data points get assigned different colours. No answer (i.e. \perp) means that some points in A remain unclassified by f .

7.2.2 A Theory of Treating Adversarial Examples

One specific verification task that has caught the attention of the ML community is to find *adversarial examples* [150, 67, 78] or to prevent them from occurring. One says that an adversarial example occurs when a ‘small’ change to the input results in an ‘unreasonably large’ change in the output (i.e. akin to our fourth task above). For example, given a correctly classified image, small, even unnoticeable changes to the pixels in the image can vastly change the classification output.

Example 7.9 In particular, image-based Deep Neural Networks (DNNs) can be easily fooled with precise pixel manipulation. The work in [166] uses a Gaussian mixture model to identify key points in images that describe the saliency map of DNN classifiers. Modifying these key points may then change the classification label made by said DNN. They explore their approach to ‘traffic light challenges’ (publicly available dashboard images of traffic lights with red/green annotations). In this challenge, they find modifying a single pixel is enough to change neural network classification.

One useful application of the map `constantValue` is using it on some *small* regions that we are interested in. In ML terms, it addresses the question if there are adversarial examples for a classifier in the vicinity of x . To characterise small regions, we would have available a metric, and then wish to use closed balls $\overline{B}(x, r)$ as inputs to `constantValue`.

To this end, we need to obtain closed balls $\overline{B}(x, r)$ as elements of $(\mathcal{V} \wedge \mathcal{K})(\mathbf{X})$. The property that for every $x \in \mathbf{X}$ we can find an $R > 0$ such that for every $r < R$ we can compute $\overline{B}(x, r) \in \mathcal{K}(\mathbf{X})$ is a characterisation of effective local compactness of a computable metric space \mathbf{X} [128]. We generally get $\text{cl}B(x, r)$, the closure of the open ball, as elements of $\mathcal{V}(\mathbf{X})$. For all but countably many radii r we have that $\overline{B}(x, r) = \text{cl}B(x, r)$, and we can effectively compute suitable radii within any interval [128].

Definition 7.10 Let (\mathbf{X}, d) be a computable metric space, and $\mathcal{C}(\mathbf{X}, \mathbf{k}_\perp)$ the space of classifiers. The map `locallyConstant` : $\mathbf{X} \times \mathbb{R}^+ \times \mathcal{C}(\mathbf{X}, \mathbf{k}_\perp) \rightarrow \mathbf{2}_\perp$ returns 1 on input (x, ε, f) iff f returns the same colour $n \in \mathbf{k}$ for all $y \in \mathbf{X}$ with $d(x, y) \leq \varepsilon$. It returns 0 if there exists some $y \in \mathbf{X}$ with $d(x, y) < \varepsilon$ such that f returns distinct colours on x and y . It returns \perp , if some points ε -close to x remain unclassified by f (such as in the cases of the global-robustness property [98]), but no distinct colours appear; or if there is a distinct colour appearing at a distance of exactly ε to x .

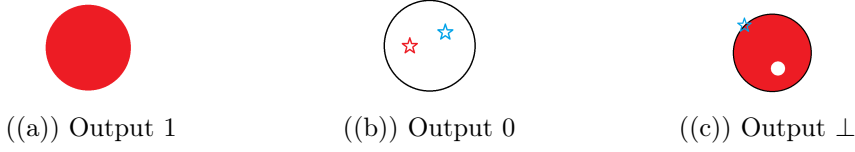


Figure 7.1: Illustrating the map locallyConstant

The map locallyConstant is illustrated in Figure 7.1. Consider the (closed) ε -ball around a point x . For a classifier f , locallyConstant outputs 1 if everything in the ball yields the same answer under f (Figure 7.1(a)). The answer is 0 if there are two points (depicted as a red and a blue star in the Figure 7.1(b)) that yield distinct answers in the ball. The answer \perp appears in two cases: If there are unclassified points inside the open ε -ball around x (the white region inside the ball on the right), or if another colour appears on the boundary of the ball, but not inside it (blue star), both shown in Figure 7.1(c).

We prove in Theorem 7.11 below that locallyConstant is computable under mild assumptions, namely that every closed ball $\overline{B}(x, \varepsilon)$ is compact (which implies that (\mathbf{X}, d) is locally compact). The Euclidean space \mathbb{R}^n is both a typical example for an effectively locally compact computable metric space where all closed balls are compact, and the predominant example relevant for ML.

Theorem 7.11 Let \mathbf{X} be an effectively locally compact computable metric space with metric d . There is a computable multivalued function producing on input $x \in \mathbf{X}$ some real $E^x > 0$ as well as an operation $\text{locallyConstant}_x : (0, E^x) \times \mathcal{C}(\mathbf{X}, \mathbf{k}_\perp) \rightarrow \mathbf{2}_\perp$, where $\text{locallyConstant}_x(r, f) = 1$ iff $\forall y \in \overline{B}(x, r) f(x) = f(y) \neq \perp$, and $\text{locallyConstant}_x(r, f) = 0$ iff $\exists y_0, y_1 \in B(x, r) \perp \neq f(y_0) \neq f(y_1) \neq \perp$.

If (\mathbf{X}, d) is such that every closed ball is compact, we can ignore the E^x and have $\text{locallyConstant} : \mathbf{X} \times \mathbb{R}^+ \times \mathcal{C}(\mathbf{X}, \mathbf{k}_\perp) \rightarrow \mathbf{2}_\perp$ instead, with $\text{locallyConstant}(x, r, f) = \text{locallyConstant}_x(r, f)$.

Proof. By [128, Proposition 12], given a point x in an effectively locally compact computable metric space, we can compute a radius E^x and the corresponding closed ball $\overline{B}(x, E^x)$ as a compact set. Subsequently, we can obtain every closed ball as a compact set given a radius less than E^x . If every closed ball is compact, we can even obtain them computably as elements of $\mathcal{K}(\mathbf{X})$ by [128, Proposition 10].

In either case, we have $\overline{B}(x, r)$ available to us as a compact set, and can thus correctly answer 1 (if this is the case) by running forallValue from Proposition 7.4 for all finitely many possible colours and $\overline{B}(x, r)$.

Without any constraints on the computable metric space \mathbf{X} we can obtain $\text{cl}B(x, r) \in \mathcal{V}(\mathbf{X})$, and we note that if there are two points assigned to distinct colours in $\text{cl}B(x, r)$, there are already such points in $B(x, r)$. Hence, running existsValue from Proposition 7.4 on all finitely many possible colours and $\text{cl}B(x, r)$ lets us correctly answer 0, if this is the case we are in. \square

To relate back to adversarial examples, `locallyConstant` tells us whether a classifier admits an adversarial example close to a given point x .

Example 7.12 In the context of fully-autonomous vehicles that use sensor-captured data as input for DNN models, [78] explains how lighting conditions and angles, as well as defects in sensor equipment themselves, yield realistic adversarial examples. Assume the metric to be chosen to model the impact of these issues on the sensor data. Then one could deploy `locallyConstant` on a fully-autonomous vehicle in order to detect if one can trust the classifier on the current input data (answer 1) or not (answer 0).

An adversarial example is the result of a small change or perturbation to the original input that results in a change of classification made by, say, a DNN. That is given the classifier f and an input x , an adversarial example is $f(x) \neq f(x + r)$ for $\|r\| \leq \epsilon$ and $\epsilon > 0$. The question is: what do we call a ‘small’ perturbation, i.e., how does one choose the parameter r ?

Example 7.13 Assume that we want to use our classifier to classify measurement results with some measurement errors. As an example, let us consider the use of ML techniques to separate Laser Interferometer Gravitational-wave Observatory (LIGO) sensor data indicating gravitational waves from terrestrial noise (e.g. [144]). If our measurements are only precise up to ϵ , then having an adversarial example for $r = \epsilon$ tells us that we cannot trust the answers from our classifier. In the example, this could mean finding that the precise values our sensors show are classified as indicating a gravitational wave, but a negligible perturbation would lead to a ‘noise’-classification.

We could use domain knowledge to select the radius r [108]. For example, in an image classification task, we could assert a priori that changing a few pixels only can never turn a picture of an elephant into a picture of a car. If we use Hamming distance as a metric on the pictures, stating what we mean with *a few pixels* gives us the value r such that any adversarial example demonstrates a fault in the classifier. Another example by [135] finds the upper and lower bounds of the input space via an optimisation procedure, following that DNNs are Lipschitz continuous functions and all values between these bounds are reachable.

So far it was the responsibility of the user to specify a numerical value for what a ‘small’ perturbation is in the definition of adversarial examples. As an alternative, we can try to compute the maximal value r such that on any scale smaller than r the point under consideration is not an adversarial example.

Proposition 7.14. Let \mathbf{X} be an effectively locally compact computable metric space with metric d such that all closed balls are compact. Then the map

$$(x, f) \mapsto \sup\{r \in \mathbb{R} \mid \exists i \in \mathbf{k} \forall y \in \overline{B}(x, r) \quad f(y) = i\} : \mathbf{X} \times \mathcal{C}(\mathbf{X}, \mathbf{k}_\perp) \rightarrow \overline{\mathbb{R}}_<$$

is computable.

Proof. As \mathbf{k} is overt and $\overline{B}(x, r)$ is uniformly compact by assumption (via [128, Proposition 10]), it follows that

$$\exists i \in \mathbf{k} \forall y \in \overline{B}(x, r) \quad f(y) = i$$

is semidecidable in the parameters x , f and r , thus

$$\{r \in \mathbb{R} \mid \exists i \in \mathbf{k} \forall y \in \overline{B}(x, r) \quad f(y) = i\} \in \mathcal{O}(\mathbb{R})$$

is an open set computable from x and f . The supremum of a set $U \in \mathcal{O}(\mathbb{R})$ is computable as an element of $\overline{\mathbb{R}}_{<}$ as shown in [126, Section 10]. \square

Proposition 7.15. Let (\mathbf{X}, d) be a computable metric space. Then the map

$$(x, f) \mapsto \inf\{r \in \mathbb{R}^{\geq 0} \mid \exists y \in B(x, r) \perp \neq f(x) \neq f(y) \neq \perp\} : \mathbf{X} \times \mathcal{C}(\mathbf{X}, \mathbf{k}_{\perp}) \rightarrow \mathbb{R}_{>}$$

is computable.

Proof. It makes no difference whether we use $\exists y \in B(x, r)$ or $\exists y \in \text{cl}B(x, r)$ here. In a computable metric space \mathbf{X} , we can compute $\text{cl}B(x, r) \in \mathcal{V}(\mathbf{X})$ from x and r . In addition, we observe that $\perp \neq f(x) \neq f(y) \neq \perp$ is semidecidable in x, y and f . Altogether, this gives us access to

$$\{r \in \mathbb{R}^{\geq 0} \mid \exists y \in B(x, r) \perp \neq f(x) \neq f(y) \neq \perp\} \in \mathcal{O}(\mathbb{R}^{\geq 0}),$$

and we can then compute the infimum in $\mathbb{R}_{>}$. Since we consider the infimum in the space $\mathbb{R}^{\geq 0}$, it is at least 0. \square

Corollary 7.16 Let \mathbf{X} be an effectively locally compact computable metric space with metric d such that all closed balls are compact. The map $\text{OptimalRadius} : \subseteq \mathbf{X} \times \mathcal{C}(\mathbf{X}, \mathbf{k}_{\perp}) \rightarrow \mathbb{R}$ defined by $(x, f) \in \text{dom}(\text{OptimalRadius})$ iff $f(x) \neq \perp$, $\exists y \perp \neq f(y) \neq f(x)$ and $\forall r, \varepsilon > 0 \exists z \in B(x, r + \varepsilon) \setminus B(x, r) \quad f(z) \neq \perp$; and by

$$\begin{aligned} \text{OptimalRadius}(x, f) &= \sup \{r \in \mathbb{R} \mid \exists i \in \mathbf{k} \forall y \in \overline{B}(x, r) \quad f(y) = i\} \\ &= \inf \{r \in \mathbb{R} \mid \exists y \in B(x, r) \perp \neq f(x) \neq f(y) \neq \perp\} \end{aligned}$$

is computable.

7.2.3 A Theory of Learners and their Robustness

Let's now delve into the process of training the classifier, considering it as a one-step procedure without adopting a dynamic view. In this context, we focus solely on supervised learning, where the dataset comprises labeled examples, and the learning algorithm's goal is to establish a function mapping feature vectors to labels. Our understanding of a learner, formalised in Definition 7.17, is based on its ability to convert finite sequences of labeled points into classifiers. Shifting from our previous emphasis on pre-trained ML procedures, we transition to discussing the essence of the learning process itself. This is where the concept of a *learner* comes into play, as it takes a set of data points paired with corresponding labels and produces a trained model.

Definition 7.17 A *learner* is a (computable/continuous) procedure that takes as an input a tuple $((x_0, n_0), \dots, (x_\ell, n_\ell)) \in (\mathbf{X} \times \mathbf{k})^*$ and outputs a classifier $f \in \mathcal{C}(\mathbf{X}, \mathbf{k}_\perp)$. The collection of all learners is the space $\mathcal{C}((\mathbf{X} \times \mathbf{k})^*, \mathcal{C}(\mathbf{X}, \mathbf{k}_\perp))$.

Note that we not only consider classifiers to be continuous functions but also assume that the learning process itself is continuous. This arises due to the requirement that learning is a computable process. In this context, continuity implies that if the data sample is modified by substituting some x_k with a closely related x'_k , the resulting classifiers f and f' cannot assign distinct colors to the same point y (although they might still vary in the use of \perp).

We do not prescribe any specific relationship between the training data and the resulting classifier's behaviour. While it might seem reasonable to expect that a learner L accurately reproduces the training data, that is, satisfies $L((x_i, n_i)_{i \leq \ell})(x_m) = n_m$, such a criterion is generally impossible to meet. This is because our concept of training data does not exclude the possibility of having multiple occurrences of the same sample point with different labels. Furthermore, this criterion would not align with practical applications, as it's often desirable for a model to be able to disregard portions of its training data as potentially flawed.

However, we can inquire whether a learner (such as a Convolutional neural network (CNN)) will produce a classifier that faithfully reproduces non-contradictory training data, assuming the following.

Proposition 7.18. Let \mathbf{X} be computably overt and computably Hausdorff. The operation

$$\text{doesDeviate} : \mathcal{C}((\mathbf{X} \times \mathbf{k})^*, \mathcal{C}(\mathbf{X}, \mathbf{k}_\perp)) \rightarrow \mathbb{S}$$

returning true on input L iff there is some input $(x_i, n_i)_{i \leq \ell} \in (\mathbf{X} \times \mathbf{k})^*$ with $x_i \neq x_j$ for $i \neq j$, and some $m \leq \ell$ such that $L((x_i, n_i)_{i \leq \ell})(x_m) \in \mathbf{k} \setminus \{n_m\}$ is computable.

Proof. If \mathbf{X} is overt, then so is $(\mathbf{X} \times \mathbf{k})^*$. Since the condition for answering true is defined by existentially quantifying a semidecidable condition over an overt set, it is semidecidable itself. \square

Robustness under additional training data Now we can ask how 'robust' the classifier we are learning with the training data actually is. This phenomenon has recently attracted attention in the ML literature under the term of underspecification [4]. Whether we view the phenomenon as robustness of learning or underspecification of the desired outcome is a matter of perspective: A failure of robustness is tied to the existence of (almost) equally good alternative classifiers.

Generally, our goal will not be so much to algorithmically verify properties of learners for arbitrary training data, but rather be interested in the behaviour of the learner on the given training data and hypothetical small additions to it. One question here would be to ask how robust a classifier is under small additions to the training data. A basic version of this would be:

Proposition 7.19. Let \mathbf{X} be computably compact and computably overt. The map

$$\text{robustPoint} : \mathbf{X} \times (\mathbf{X} \times \mathbf{k})^* \times \mathcal{C}((\mathbf{X} \times \mathbf{k})^*, \mathcal{C}(\mathbf{X}, \mathbf{k}_\perp)) \rightarrow \mathbf{2}_\perp$$

answering 1 on input $x, (x_i, n_i)_{i \leq \ell}$ and L iff

$$\forall x_{\ell+1} \in \mathbf{X} \forall n_{\ell+1} \in \mathbf{k} \quad L((x_i, n_i)_{i \leq \ell})(x) = L((x_i, n_i)_{i \leq \ell+1})(x) \in \mathbf{k}$$

and answering 0 iff

$$\exists x_{\ell+1} \in \mathbf{X} \exists n_{\ell+1} \in \mathbf{k} \quad \perp \neq L((x_i, n_i)_{i \leq \ell})(x) \neq L((x_i, n_i)_{i \leq \ell+1})(x) \neq \perp$$

is computable.

Proof. Since \mathbf{X} and \mathbf{k} are both compact and overt, both universal and existential quantification preserves semidecidable predicates. Since \mathbf{k} is discrete and Hausdorff, the stem of our predicate definitions for answering 0 and 1 are both semidecidable. \square

We can lift `robustPoint` to ask about all points in a given region, or even in the entire space as a corollary:

Corollary 7.20 Let \mathbf{X} be computably compact and computably overt. The map

$$\text{robustRegion} : (\mathcal{K} \wedge \mathcal{V})(\mathbf{X}) \times (\mathbf{X} \times \mathbf{k})^* \times \mathcal{C}((\mathbf{X} \times \mathbf{k})^*, \mathcal{C}(\mathbf{X}, \mathbf{k}_\perp)) \rightarrow \mathbf{2}_\perp$$

answering 1 on input $A, (x_i, n_i)_{i \leq \ell}$ and L iff `robustPoint` answers 1 for every $x \in A$ together with $(x_i, n_i)_{i \leq \ell}$ and L , and which answer 0 iff there exists some $x \in A$ such that `robustPoint` answers 0 on input $x, (x_i, n_i)_{i \leq \ell}$ and L , and which answers \perp otherwise, is computable.

If \mathbf{X} is computably compact and computably overt, and we take the regions A to be themselves compact and overt sets, then both operations are computable.

Corollary 7.21 The map

$$\text{globallyRobust} : (\mathbf{X} \times \mathbf{k})^* \times \mathcal{C}((\mathbf{X} \times \mathbf{k})^*, \mathcal{C}(\mathbf{X}, \mathbf{k}_\perp)) \rightarrow \mathbf{2}_\perp$$

defined as

$$\text{globallyRobust}((x_i, n_i)_{i \leq \ell}, L) = \text{robustRegion}(X, (x_i, n_i)_{i \leq \ell}, L)$$

is computable.

7.2.4 Sparsity of Training Data

Allowing arbitrary additional training data as in the definition of robustness might not be too suitable – for example, if we add the relevant query point together with another label to the training data, it would not be particularly surprising if the new classifier follows the new data. If we bring in a metric structure, we can exclude new training data which is too close to the given point.

Definition 7.22 Fix a learner $L : (\mathbf{X} \times \mathbf{k})^* \rightarrow \mathcal{C}(\mathbf{X}, \mathbf{k}_\perp)$, some $N \in \mathbb{N}$ and $\varepsilon > 0$. We say that $(x_i, n_i)_{i \leq \ell}$ is *sparse* at $x \in \mathbf{X}$, if there are $(y_i, m_i)_{i \leq j}$ and $(y'_i, m'_i)_{i \leq j'}$ such that $\ell + N \geq j, j' \geq \ell$, $y_i = y'_i = x_i$ and $m_i = m'_i = n_i$ for $i \leq \ell$, and $d(y_i, x), d(y'_i, x) > \varepsilon$ for $i > \ell$ satisfying $\perp \neq L((y_i, m_i)_{i \leq j})(x) \neq L((y'_i, m'_i)_{i \leq j'})(x) \neq \perp$.

We say that $(x_i, n_i)_{i \leq \ell}$ is *dense* at $x \in \mathbf{X}$ if for all $(y_i, m_i)_{i \leq j}$ and $(y'_i, m'_i)_{i \leq j'}$ such that $\ell + N \geq j, j' \geq \ell$, $y_i = y'_i = x_i$ and $m_i = m'_i = n_i$ for $i \leq \ell$, and $d(y_i, x), d(y'_i, x) \geq \varepsilon$ for $i > \ell$ it holds that $L((y_i, m_i)_{i \leq j})(x) = L((y'_i, m'_i)_{i \leq j'})(x) \neq \perp$.

To put it in words: Training data is dense at a point whose label it determines, even if we add up to N additional points to the training data, which have to be at least ε away from that point. Conversely, at a sparse point, we can achieve different labels by such an augmentation of the training data. If we have chosen the parameters N and ε well, then we can conclude that based on the training data we can make reasonable assertions about the dense query points, but unless we have some additional external knowledge of the true distribution of labels, we cannot draw reliable conclusions about the sparse query points. We concede that it would make sense to include points under *sparse* where the classifiers will always output \perp even if we enhance the training data, but this would destroy any hope of nice algorithmic properties.

Theorem 7.23 Let \mathbf{X} be a computably compact computable metric space. The operation

$$\text{SprsOrDns} : \mathcal{C}((\mathbf{X} \times \mathbf{k})^*, \mathcal{C}(\mathbf{X}, \mathbf{k}_\perp)) \times \mathbb{N} \times \mathbb{R}_+ \times (\mathbf{X} \times \mathbf{k})^* \times \mathbf{X} \rightarrow \mathbf{2}_\perp$$

answering 0 on input $L, N, \varepsilon, (x_i, n_i)_{i \leq \ell}$ and x iff $(x_i, n_i)_{i \leq \ell}$ is sparse at x , and answers 1 if $(x_i, n_i)_{i \leq \ell}$ is dense at x is computable.

Proof. Essentially, the claim is that both being sparse and being dense is a semidecidable property. We show this by examining Definition 7.22 and in particular its quantification structure.

We note that $\perp \neq L((y_i, m_i)_{i \leq j})(x) \neq L((y'_i, m'_i)_{i \leq j'})(x) \neq \perp$ is a semidecidable property. We are then adding existential quantifications over $\text{cl}\{y \in \mathbf{X} \mid d(x, y) > \varepsilon\}$, and this set is always available to us as an overt set when working in a computable metric space. Thus, being sparse is semidecidable.

For density, note that $L((y_i, m_i)_{i \leq j})(x) = L((y'_i, m'_i)_{i \leq j'})(x) \neq \perp$ is again semidecidable. This time we add universal quantification over $\{y \in \mathbf{X} \mid d(x, y) \geq \varepsilon\}$. This set is available to us as a compact set thanks to the demand that \mathbf{X} be a computably compact computable metric space. \square

Figures 7.2 and 7.3 illustrate the concepts of robustness and sparsity/density using the same dataset ¹. In Figure 7.2, we present a classification example where our algorithm employs a SVM to establish a separation line between gray and red dots. The left side

¹Our overall algorithms are theoretical and not easily translated into code. However, the individual concepts can be implemented in code to aid the comprehension of mathematical ideas like robustness/sparsity and density as seen in Appendix A.

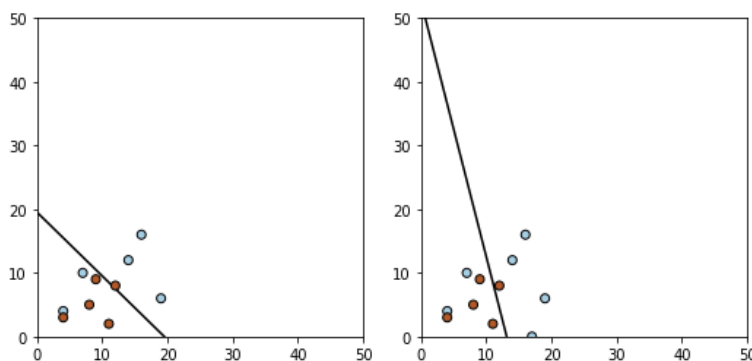


Figure 7.2: Illustrating lack of robustness. Left: Original data, Right: Changed separating line due to one added data point (at $(18,0)$).

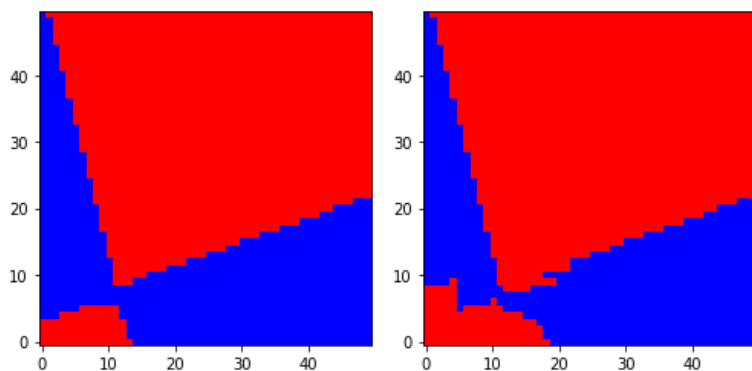


Figure 7.3: Illustration of predicates concerning the original classifier. Left: Robustness (Red: Robustness, Blue: Not Robust), Right: Sparsity/Density (Red: Dense, Blue: Sparse) – note: \perp is at the boundaries between the coloured areas.

displays the original dataset, while the right side depicts the dataset after adding one more gray data point. Notably, the separation line has shifted significantly.

In the left image of Figure 7.3, one can observe the regions within the input space that could experience classification changes due to the addition of a single point to the training data. The algorithm introduces the extra data point, computes the separation line, and evaluates the position of each point in relation to this line. This process is repeated for all potential additional data points, revealing the points that remain consistently on one side of the line (robust) and those that do not. The blue region encompasses points that lack robustness, meaning they may switch sides of the separation line depending on the supplementary point. The red region includes robust points, which consistently remain on the same side of the separation line, regardless of the additional points.

Moving to the right image of Figure 7.3, we introduce the concept of sparsity/density. Similar to before, we examine whether a point could change sides of the separation line with the addition of another sample point. However, our focus is solely on sample points

located more than 5 units away from the point under consideration. It’s possible for points to be dense but not robust (as seen in the area directly above $(10, 0)$ and $(20, 0)$). This can be clarified as follows: to achieve a steep separation line, the extra data point must be placed around $(15, 0)$. Consequently, the steep separation lines contribute to the upper pattern in the image without affecting the red component at the bottom.

Density implies robustness, but the reverse is not necessarily true, as density encompasses additional training data that isn’t too close to the point under consideration. The choice between these notions depends on the specific application. A lack of robustness when a single data point is added signals an inadequately sized training dataset. However, if a more substantial portion of supplementary training data is introduced, demanding robustness might become excessive. In such instances, dense points remain those where only exceedingly similar counterexamples would contest the classifier’s response.

7.2.5 Sparsity in the ML literature

In the context of this study, the term ‘sparsity’ pertains to the *measurement* of the input representation, rather than denoting the representation itself (such as in sparse matrix forms). Specifically, we can assess the *sparsity* of sampling from any underlying manifold, which may subsequently impact how ML algorithms learn and formulate classifications within these specific spatial regions.

In regions of a manifold characterised by sparsity, the available information for ML algorithms to learn from is limited. Such regions might consequently accommodate a higher abundance of adversarial examples [100]. These measurement techniques can potentially aid in the generation or search for adversarial examples by manipulating the size of the closed ball $\overline{B}(x, r)$ in relation to the local sparsity information of the manifold’s sampling.

A common approach to measuring the sparsity of sampling within data employs kernel density estimation (KDE), often utilising a Gaussian kernel such as $\exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$, where σ controls the kernel’s width. For low σ values, the similarity matrix forms an identity matrix, while high values yield a matrix of all ones. Thus, the optimal bandwidth parameter choice is context-specific, requiring tuning to effectively quantify the manifold space’s sparsity.

However, efforts have been made to address this challenge. The work in [100] outlines two methods for adaptively setting bandwidth parameters in KDE, tailoring them to the local characteristics of regions. This approach optimally captures the sparsity measurement with respect to each manifold space sample.

7.2.6 Linear Regression

Regression models are supervised learning models that contain past data with labels which are then used to build the model, where the predicted output variable is continuous in nature. Linear regression assumes a linear relationship between two variables, the independent and dependent, and aims to find the best-fitting line to describe the relationship.

Definition 7.24 (Linear Regression) Given finitely many points $(x_i, y_i)_{i \leq n}$ in \mathbb{R}^2 , find some a, b that minimise the summed squared errors $\sum_{i \leq n} (ax_i + b - y_i)^2$.

Linear regression is computable on ‘reasonable data’. The precise requirement is that there exist values i and j such that $i, j \leq n$ and $x_i \neq x_j$ (or when $n = 1$). Without this constraint, numerous solutions (a, b) are possible, which subsequently leads to non-computability.

Proposition 7.25. Finding the line of best fit using linear regression is Weihrauch equivalent to LPO.

If we are dealing with reasonable data, we can perform linear regression in the ‘normal’ manner. In cases where we encounter identical x values, we can calculate the average of all corresponding y values and derive a line of best fit based on this averaged data. The slope of this line provides insight into the relationship between y values and x values close to 0. In scenarios where we lack knowledge of the exact line parameters but possess the function, we can determine the slope by examining the values at 0 and 1.

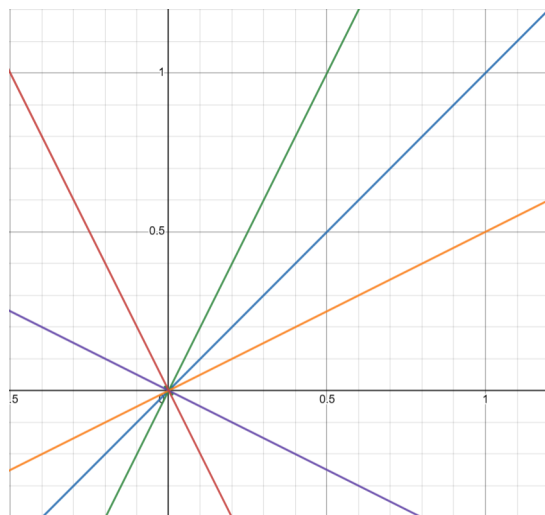


Figure 7.4: Two points at $(0,0)$. The line can be anywhere.

For instance, when we have two points located at $(0,0)$, the relationship between x and y remains undefined. Consequently, the line could encompass any of the possibilities depicted in Figure 7.4, or it could essentially be any line passing through $(0,0)$. This scenario illustrates an instance where the input of LPO consists of a binary sequence containing only 0s.

If we consider a point at $(0,0)$ and another at $(1,1)$, the line of best fit is simply $y = x$, as depicted in Figure 7.5(a). As the subsequent x and y points draw closer to the origin $(0,0)$, the slope of the line becomes steeper, demonstrated in Figure 7.5. This characteristic mirrors the timing of the occurrence of the ‘1’ in the input of LPO; the later this ‘1’ appears, the more pronounced the steepness of the line.

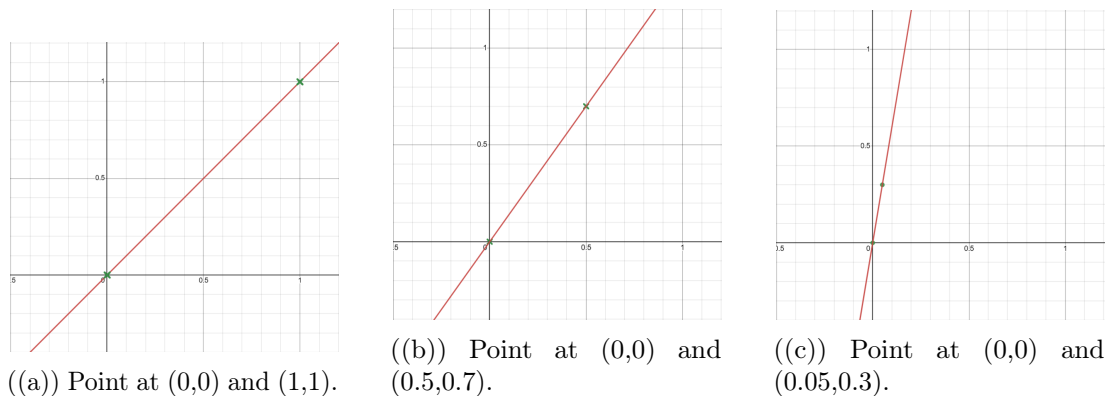


Figure 7.5: The later a 1 occurs in the input of LPO, the steeper the line becomes.

7.3 Related Work

In 2010, Pulina and Tachella introduced an approach for verifying linear arithmetic constraints in multiplayer perceptions by translating them into SAT instances [132]. Fast forward a decade, a comprehensive systematic review of neural network testing and verification covered an impressive 91 articles [168]. A dedicated survey specifically centered around the verification of deep neural networks can be found in [101]. In this context, we direct our attention to the operation referred to as `forallValue` (as detailed in Proposition 7.4) and its broader applicability beyond classification tasks. What sets our approach apart is its model-agnostic nature. It remains independent of minor specifics, including choices like the selection of activation functions.

When it comes to neural networks, a broader approach towards the decidability of verification questions begins with the observation that, given the utilisation of piecewise linear activation functions and specifications that can be defined in the theory of real closed fields (i.e., quantified formulas involving $+$, \times , and \leq), we inherently achieve decidability. This signifies a binary outcome (yes/no answers) without the need for \perp . This stems from the theory of real closed fields and although it is theoretically decidable, it often comes with infeasible computational complexity. This insight has been pointed out in prior works, such as [80].

Recently, by employing probably approximately correct (PAC) learning theory (for background, refer to [142]), a study delving into the intermediate scenario emerged, where learners are required to be computable without the constraint of resource bounds. This achievement is attributed to [2]. In their work, they established a concept akin to our notion of a computable learner. Their approach leveraged key principles of a computably enumerable representable (CER) hypothesis class, combined with an empirical risk minimisation (ERM) learner. This enabled them to identify an ERM learner that maintains computability across every CER class that is PAC learnable within the realisable case. It's important to highlight, however, that the study in [2] does not delve into verification questions as part of its focus.

7.4 Conclusions

We motivated and presented a number of questions that one might want to ask when verifying classifiers obtained by ML. These include elementary questions such as whether any point in a region gets assigned a particular colour, but also more advanced ones such as whether adversarial examples exist. Finally, we make a contribution to the phenomenon of underspecification by studying the robustness of learners. Using the framework of computable analysis we are capable of precisely formalising these questions, and proving them to be computable under reasonable (and necessary) assumptions.

Regarding the necessity of the assumptions, we point out that dropping conditions or considering maps providing more information instead, will generally lead to non-computability. We leave the provision of counterexamples, as well as potentially a classification of *how* non-computable these maps are to future work. The notion of a *maximal partial algorithm* recently proposed by Neumann [113] also seems a promising approach to proving the optimality of our results.

There is a trade-off between the robustness of a classifier and its ‘accuracy’. It seems possible to develop a computable quantitative notion of robustness for our function `locallyConstant`, which could then be used as part of the training process in a learner. This could be the next step to adversarial robustness [37, 67].

Rather than just asking questions about particular given classifiers or learners, we could start with a preconception regarding what classifier we would want to obtain for given training data. Natural algorithmic questions then are whether there is a learner in the first place that is guaranteed to meet our criteria for the classifiers and whether we can compute such a learner from the criteria.

Chapter 8

Hypergames and Cybersecurity

Contents

8.1	Hypergames and Attack Graphs	95
8.2	Applications to Cybersecurity	103
8.3	Suggested Research Directions	110
8.4	Related Research Areas	114
8.5	Conclusions	116

This chapter is a collection of different projects. This includes work converting cyber attack graphs to hypergames which is joint work with Martin Barrere Cambrun and Chris Hankin (Section 8.1). There is an introduction to using regret minimisation as a solution concept for hypergames which is joint work with Arno Pauly (Section 8.1.3). The remainder of the chapter is based on a literature review paper ‘Hypergames, Cybersecurity and RL’ that is pending publication. The lead author is the researcher with joint work with Andrew Fielder, Paul Jones, and Conor Artman.

8.1 Hypergames and Attack Graphs

Cybersecurity scenarios often manifest as games involving simultaneous moves, commonly referred to as Nash games. These games frequently fall into the realm of imperfect complete information games, wherein players lack awareness of each other’s actions. Nevertheless, all game-related information is widely shared knowledge, thereby ensuring no information advantage.

However, these conventional games don’t offer a fully realistic depiction of real-world scenarios. In certain instances, attackers possess system ‘backdoors’, allowing them to operate undetected by defenders. In such cases, defenders are unaware of the intruders’ movements and cannot preemptively counter their system entry. Yet, defenders still aim to optimise their defensive strategies. This is precisely where hypergames, along with their model of incomplete information, come into play as a valuable tool.

In hypergames, this model of incomplete information proves advantageous for addressing scenarios where the presence of hidden backdoors challenges the traditional assumptions of simultaneous moves and shared information.

8.1.1 Attack Graphs

Attack graphs are a visual and analytical tool used in the field of cybersecurity to model and understand the potential paths that attackers could take to compromise a computer system or network. They help security analysts identify vulnerabilities, potential attack vectors, and the possible consequences of successful attacks.

In essence, an attack graph is a graphical representation that maps out the different steps an attacker might take to reach a specific goal, such as gaining unauthorised access to a system or extracting sensitive information. Each step in the graph represents a specific action or attack, and the edges between steps represent the dependencies and conditions that must be met for one attack to lead to another.

Each node in the graph represents a state or condition of the system. This could be a configuration setting, a service, a user account, or any other element that is relevant to the security of the system. Edges between nodes represent the possible attack paths an attacker could follow. An edge might indicate that an attacker needs to compromise one element before being able to move on to the next. For example, an attacker might need to gain access to a user account before being able to escalate privileges.

Each node can have associated attack steps. These are the specific actions an attacker could take to compromise that element. For instance, compromising a user account might involve guessing the password or exploiting a vulnerability in the login process. Attack graphs often include conditions that must be met for an attack step to be successful. For example, an attacker might need to have gained access to a specific network segment before attempting to exploit a vulnerability on a particular server.

By analysing the attack graph, security analysts can assess the potential risk and impact of different attack scenarios. This helps them prioritise which vulnerabilities and paths need to be addressed more urgently.

Attack graphs are particularly useful for modeling complex systems where there are multiple paths an attacker could take to achieve their objectives. They provide a structured way to visualise and understand the interactions between various elements in the system, helping security teams develop effective defense strategies, prioritise patches and updates, and enhance overall system security.

8.1.1.1 AND/OR Graphs

AND/OR graphs are another type of graphical representation used in the context of cybersecurity, particularly for modelling and analysing security scenarios. These graphs are employed to represent the relationships between attack steps, dependencies, and possible outcomes in a more structured and organised manner. Simple AND/OR graphs can be seen in Figure 8.1.

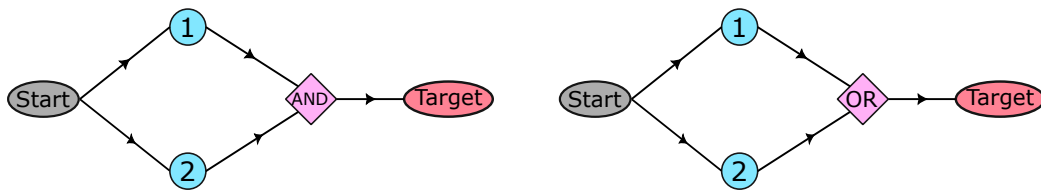


Figure 8.1: Simple ‘AND’ and ‘OR’ Graphs

In an AND/OR graph, AND nodes represent situations where multiple conditions must be satisfied simultaneously for an attack to proceed. These conditions could be multiple vulnerabilities that an attacker needs to exploit in combination to advance in the attack path. This means both nodes 1 and 2 need to be satisfied for the attacker to reach the target in Figure 8.1.

OR nodes, on the other hand, signify points in the graph where an attacker has multiple options or paths to choose from. The attacker only needs to successfully complete one of the listed options to continue their progression. This means either nodes 1 or 2 needs to be satisfied for the attacker to reach the target.

These two types of nodes, AND/OR, allow for a more nuanced representation of attack scenarios and dependencies. Attack steps and conditions can be organised in a way that reflects the complexity of real-world attacks.

8.1.2 Converting Attack Graphs into Hypergames

In order to transform attack graphs into a hypergame, we need to establish the players, strategies, utilities, and payoffs. Let’s explore a two-player game with an attacker and a defender¹.

8.1.2.1 Defenders Perspective

The attack graph in Figure 8.2 models how the defender comprehends the ways in which an attacker can compromise a database server within their network. They are only aware of two hosts that the attacker could exploit.

We need to establish the strategies of each player. For the defender we allow them to ‘remove’ an edge of the graph, which prevents the attacker from progressing, or reaching that host. This means the defender has three strategies, to block access to Host1, Host2 or the Target.

As for the attacker, their objective is to reach the target, and in this instance, they have two routes. They can go through Host1 or Host2, both of which allow them to compromise the target. Therefore, the attacker has two strategies, denoted by the vulnerabilities they will exploit: V1V3 or V2V3.

¹Thanks to Martin Barrere Cambrun and Chris Hankin for providing the attack graphs and project.

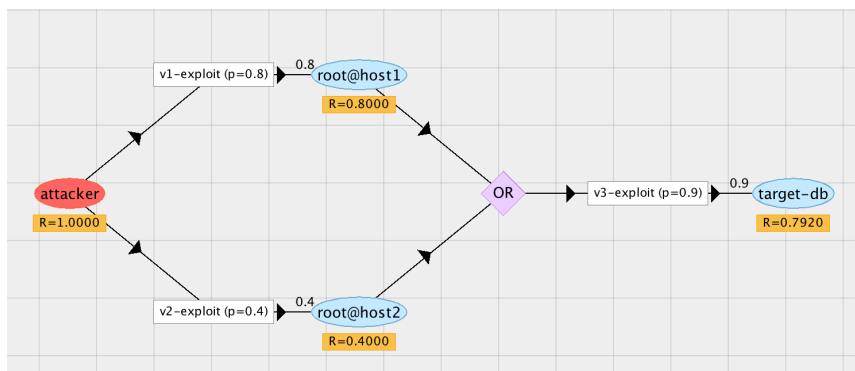


Figure 8.2: The Attack graph from the Defenders Perspective

To calculate the final payoffs for the attacker, we must determine the estimated payoff if the target is compromised via a specific route and the associated cost for the attacker. For this example, we will use arbitrary numbers for the costs and expected payoffs. However, we will utilise the probabilities of a successful attack from the graph itself. The attackers and defenders will have distinct values for these factors, so we will outline them individually below.

Node	Expected Payoff	Probability of Success	Cost
V1	3	0.8	4
V2	10	0.4	6
V3	10	0.9	3

Table 8.1: Attackers Node Payoffs, Probabilities and Cost

The payoff the attacker receives at the end of the game depends on the expected payoff and the probability of success. For example, node V1 has a final payoff of 2.4 because $3 \times 0.8 = 2.4$. Moreover, the payoff for the strategy V1V3 is calculated by the addition of the payoff of both V1 and V3, $2.4 + 9 = 11.4$. Additionally, the attacker's overall cost for each strategy is the sum of the costs of the nodes they need to exploit.

Strategy	Final Payoff	Final Cost
V1V3	11.4	7
V2V3	13	9

Table 8.2: Attackers Strategies Payoffs and Costs

For the defenders, to simplify this game, they can only defend one node at a time. Thus, they can defend Host1, Host2, or the Target. This implies that they only need the expected payoffs for the individual nodes, as well as the cost associated with defending them.

Node	Expected Payoff	Cost
Host1	6	5
Host2	7	6
Target	10	9

Table 8.3: Defenders Node, Payoffs and Cost

The payoffs and strategies can be seen in the left-hand normal form Game in Figure 8.3. The rows are the defenders strategies, the columns are the attackers. The payoffs in each box relate to the payoff for the (defender,attacker) if both strategies are played.

		Attacker	
		V1V3	V2V3
Defender	Host1	18,-4	1,4
	Host2	1,4.4	17,-6
	Target	8,-4.6	7,-5

		Attacker		
		V1V3	V2V3	V10V3
Defender	Host1	27,-4	10,4	8,4.5
	Host2	10,4.4	26,-6	7,4.5
	Host3	9,4.4	8,4	25,-5
	Target	17,-4.6	16,-5	14,-4.5

Figure 8.3: The hypergame from the Defenders Perspective (left) and the Attacker Perspective (right).

These payoffs are created using the costs and expected values. Here are a few examples:

- If the attacker plays V1V3 and the defender plays Host1, then the attacker gets nowhere. Therefore, the attackers payoff would be the cost of V1: -4 . Then the defender would be successful and their payoff would be the expected value of all uncompromised nodes - cost $6 + 7 + 10 - 5 = 18$.
- If the attacker plays V1V3 and the defender plays Host2, then the attacker would successfully reach the target. Therefore, the payoff for the attacker would be the expected value times the probability of success minus the cost $= 11.4 - 7 = 4.4$. The defender would not be successful in stopping the attack, so their payoff would be the expected value of all uncompromised nodes minus the cost of their strategy, hence $7 - 6 = 1$.
- If the attacker plays V1V3 and the defender plays Target, then the attacker would not be able to reach the target, but would have been in the system. The attacker would then have successfully attacked V1 but missed the target making their payoff $2.4 - 7 = -4.6$. This makes sense because it would cost a lot in order to hack in but overall they have missed their target. The defender would be successful,

therefore their payoff would be the the expected value of all uncompromised nodes minus the cost of defending the target $7 + 10 - 9 = 8$.

8.1.2.2 Attacker Perspective

The attack graph models how the attacker perceives the database server and the methods to compromise it. As depicted in Figure 8.4, there exists a ‘backdoor’ route through Host3, which can be exploited using vulnerability V10.

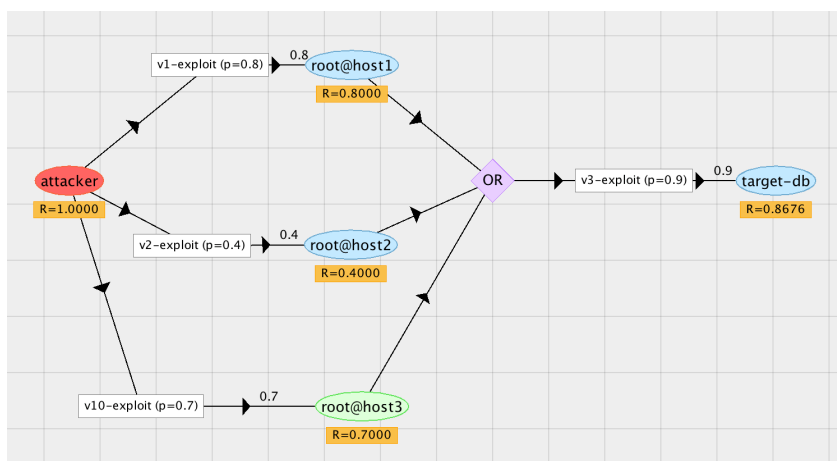


Figure 8.4: The Attack graph from the Attackers Perspective

This game is an extension of the defenders’ game shown in Figure 8.2. It includes the attacker’s strategy V10V3 and the defender’s strategy Host3. As a result, we expand Tables 8.1 8.2 and 8.3 to incorporate these new additions.

Node	Expected Payoff	Probability of Success	Cost
V1	3	0.8	4
V2	10	0.4	6
V3	10	0.9	3
V10	5	0.7	5

Table 8.4: Attackers Node Payoffs, Probabilities and Cost

Strategy	Final Payoff	Final Cost
V1V3	11.4	7
V2V3	13	9
V10V3	12.5	8

Table 8.5: Attackers Strategies Payoffs and Costs

Node	Expected Payoff	Cost
Host1	6	5
Host2	7	6
Host3	9	7
Target	10	9

Table 8.6: Defenders Node, Payoffs and Cost

The costs and expected values are calculated similarly to Section 8.1.2.1. You can observe these in the right-hand normal form Game depicted in Figure 8.3.

Further work has been achieved on this by a Masters student Soural Dandothi [52]. Arno Pauly and the researcher supervised his project which used this Section as a starting basis. The researcher provided the conversion of attack graphs to hypergames and shared the attack graph json files provided by Martin Barrere Cambrun and Chris Hankin. His work expanded this project by creating the code required to convert these json files straight into hypergames using the above method. He successfully achieved this as well as using some of the researcher’s code (Appendix B) in order to check the Hyper Nash equilibrium and Stable Hyper Nash equilibrium for the overall games.

The solution presented in Section 3.4 regarding Nash equilibria does not apply to this situation, as Nash equilibria do not exist in either the defenders’ game or the attackers’ game. This is often the case in most real-world scenarios because the concept of Nash equilibrium seeks to find a mutually optimal response for both players, which doesn’t align with the divergent objectives of the attacker and defender.

In order to make these games more realistic, the players could adopt mixed strategies. Therefore, the defender would have different probabilities of engaging with each host. This would mean that they are unsure which host the attacker will be targeting; as such, they would randomly choose a host to defend.

8.1.3 Using Regret Minimisation

A different approach to find a solution to hypergames within cybersecurity is regret minimisation. This strategy seeks to minimise the gap between the outcomes achieved through a player’s chosen actions and those achievable by different strategies. Regret minimisation strives to refine decision-making strategies by learning from past actions and their outcomes. In the context of hypergames, regret minimisation involves making decisions that minimise the potential regret associated with those decisions. Players aim to choose strategies that lead to outcomes with lower regret values, ensuring that they don’t significantly regret their choices when comparing them to alternative strategies. There can be multiple layers of interaction and evolving strategies, therefore, regret minimisation becomes more complex. Players need to consider not only their immediate regret in a specific subgame but also the potential impacts of their decisions on overall outcomes if there are moves they are unaware of. This extends the traditional concept of regret minimisation to a more dynamic and interconnected setting.

In joint work with Arno Pauly we started looking into achieving this solution in regards to hypergames. To do this we redefine a game using sources, sinks and non-increasing functions.

- Definition 8.1**
1. A D -game consists of a DAG (V, E) with unique source s and sink t ; and furthermore for each vertex $v \in V$ a non-increasing continuous function $\iota_v : [0, 1] \rightarrow [0, 1]$.
 2. A strategy for defender is an assignment $\sigma : V \rightarrow [0, 1]$.
 3. A strategy for attacker is a subset $A \subseteq V$.
 4. The *success probability* $p(\sigma, A)$ of playing σ against A is computed as follows. A vertex $v \in V$ is *attacked* if $v \in A$ and either $v = s$ or a predecessor of v is already *compromised*. If v is attacked, then it is comprised with probability $\iota_v(\sigma(v))$. The success probability is the expected probability for t to be NOT compromised.
 5. The payoff of σ against A is $p(\sigma, A) - \sum_{v \in V} \sigma(v)$.

The idea is that $\sigma(v)$ denotes the investment the defender makes into the security of node v , and ι_v tells us what the success chance of an attack on v is given a certain level of investment.

We do not assign payoffs to the attacker, as our approach does not require us to model the attacker as a rational agent.

Observation 8.2. For each strategy of the attacker, there is a best response by the defender.

Proof. Payoffs are continuous, and the strategy space of the defender is compact. \square

Definition 8.3 An extension of a D -game $(V, E, (\iota_v)_{v \in V}, s, t)$ is another D -game $(V', E', (\iota'_v)_{v \in V'}, s, t)$ with the same source and target vertex such that $V \subseteq V'$, $E \subseteq E'$, $\iota'_v = \iota_v$ for $v \in V$, and ι'_v is constant for $v \notin V$.

Note that defender strategies in a D -game naturally extend to strategies in any extension of it by setting $\sigma(v) = 0$ for any $v \notin V$. Strategies of this form are the only best responses; and we will thus ignore all other defender strategies in the extension.

Definition 8.4 Given a defender strategy σ in a D -game G , some extension G' of the D -game and an attacker strategy A in the extension, the *regret* for defender $R(\sigma, G', A)$ is the difference between the payoff achieved by defender playing a best response to A in G' and the payoff obtained by playing σ against A in G' .

Conjecture 8.5. For any D -game G there are finitely many extensions $(G'_i)_{i \in I}$ and finitely many attacker strategies $(A_i)_{i \in I}$ such that for every defender strategy σ in G , every arbitrary extension G'' of G and every attacker strategy A'' in G'' it holds that

$$R(\sigma, G'', A'') \leq \max_{i \in I} R(\sigma, G'_i, A_i)$$

Assuming that the preceding conjecture holds, we call $\max_{i \in I} R(\sigma, G'_i, A_i)$ the maximum regret for σ , and denote it by $\mathfrak{R}(\sigma)$.

Observation 8.6. $\mathfrak{R}(\sigma)$ (if well-defined) depends continuously on σ , thus for every D -game there exists a strategy σ_0 minimising $\mathfrak{R}(\sigma_0)$, i.e. satisfying $\mathfrak{R}(\sigma_0) \leq \mathfrak{R}(\sigma)$ for all strategies σ .

Conjecture 8.7. The regret-minimising strategy σ_0 is unique.

If the two conjectures are indeed true, then the unique regret-minimising strategy σ_0 would seem like a good candidate for rational behaviour when faced with “unknown unknowns” regarding potential attacks.

8.2 Applications to Cybersecurity

8.2.1 Current Applications

8.2.1.1 Cyber Attack and Defence

House and Cybenko started to apply hypergames to cyber attack and defence [77]. They achieved this by establishing Nash Equilibria Mixed Strategies (NEMS) for both the attacker and defender in each subgame of the hypergame. However, the attacker might lack knowledge of the probabilities associated with each subgame. To address this, House and Cybenko proposed two methods for learning subgame probabilities through repeated play; maximum entropy and subgame transitions as Hidden Markov Models (HMM). The underlying assumption is that the defender always plays their Nash equilibrium strategy. Both methods can be used in concert and work well together, however, the attacker’s strategy significantly influences the accuracy of the methods. Furthermore, the defender could potentially enhance their ability to conceal their subgame probabilities.

Maximum entropy allows the attacker to end up with estimates for the probabilities by the attacker computing the NEMS and utilities for the defender in each subgame. For each row/strategy, the variance is calculated as a measure of the dispersion induced by the defender switching from one subgame to another. The strategy exhibiting the highest variance is then played against the defender multiple times and the utility is recorded from each play of the game. To estimate each probability, the maximum entropy principle is employed using the Lagrange multiplier method.

HMMs can effectively illustrate transitions between subgames, using the Baum-Welch algorithm. The emissions of the HMM correspond to the utility obtained from a fixed strategy, a value that the attacker is aware of. To minimise subgame ‘overlap,’ rational assumptions are often required, alongside the assumption that the defender consistently adheres to their Nash equilibrium. By minimising this overlap, the strategy’s efficacy increases, as the utility corresponds to a single possible subgame. One approach would be to remap the utilities in order for the elements with the swiftest and most accurate learning to be the highest for the attacker. Consequently, the Nash equilibrium would represent the attacker’s best learning against the defender’s obfuscation. This method

can be employed to ascertain a ‘learning vs obfuscation’ Nash equilibrium, by the attacker learning which subgame the defender is not playing (through repeated plays) and eliminating it. This equilibrium could offer an improvement in learning to the attacker or an improvement in obfuscation to the defender.

In analysing games between attackers and defenders, Stochastic Petri Nets (SPNs) have been applied [42]. SPNs use either Markov or Semi-Markov models where time follows Poisson distribution or is exponentially distributed in relation to event arrival or inter-arrival times. A deception game can be modelled using SPNs in order to capture measures in order to observe the behaviours of the attackers and defenders.

Hypergames can effectively model the strategic interactions between attackers and defenders, making them applicable to a wide range of information security challenges. These can be achieved using a two-player first or level-2 hypergame. Analysing this requires investigating each of the individual games for stability and finding stable equilibria for the hypergame [79]. First, the attacker’s level-1 game is analysed and the set of equilibrium points of the hypergame is completed. The same procedure is then performed for the defender’s game. Finally, the level-2 hypergame is analysed as a whole in order to compute the full set of equilibrium points of the game, which is at the intersection of the sets of stable outcomes.

8.2.1.2 Optimisation

Cyber-physical systems are mechanisms designed to be controlled or monitored by computer-based algorithms. Bakker explores hypergames and cyber-physical security for control systems in order to provide another layer of robustness [7]. Their study delves into various aspects, including static problems, dynamic optimisation, and computational implementation.

A static optimisation problem was formulated and constrained within an operating envelope. This investigation led to the exploration of two scenarios: objective function manipulations and constraint manipulation. The first involves the attacker manipulating the defender’s observation of the objective function parameters by using a deterministic strategy. This results in a level-2 hypergame, if the defender knows about the attacker it advances to a higher-level hypergame. On the other hand, constraint manipulation entails the attacker altering the constraint to be more restrictive, whether that would be the real constraint or the defender’s perception. For instance, the attacker could induce the defender to significantly deviate from the operating system’s constraints to provoke a failure.

Dynamic optimisation links hypergames with the Model Predictive Control (MPC) problem. The objective is to minimise a cost function subject to state dynamics constraints and operational constraints. These result in level-2 hypergames. The study examined both static and dynamic approaches where the defender’s optimisation is based on perceived values rather than real values. The authors concluded that manipulating constraints can be a more effective attacker strategy than manipulating objective functions directly.

8.2.1.3 Metagames

Hypergames can be expanded further into Metagames by producing a hypergame over hypergame solutions. These use randomisation over reverse engineering strategies. Metagames can be used in a single-stage approach for deception-robust control. This approach has been compared to minmax hypergames within [9]. Minmax hypergames allow the defender to re-optimize the entire system in a deterministic fashion. Minmax hypergames are guaranteed to provide an upper bound at least as good as a metagame solution, under the correct assumptions. However, the metagame method is computationally cheaper in higher level games. Overall, it shows that both metagame and minmax hypergames provide performance guarantees and minimal upper bounds on the attacker's performance given the information available to the defender.

8.2.1.4 Graph Based Applications

An Attack-Defend (AD) game on a graph combines the attack graph model with the defence actions available to the defender. An AD game is a tuple $\mathcal{G} = \langle \mathcal{G}, \varphi \rangle$ where; \mathcal{G} is a Game Arena and φ is the Boolean payoff function. AD games can be integrated with hypergames to develop strategies that guarantee provable security in defense scenarios [90]. For instance, situations involving the attacker's misperceptions of payoffs due to the presence of decoy systems can be effectively addressed using sure-winning strategies within the hypergame framework.

A zero-sum AD game can be used to model a situation where the attacker's goal is specified in linear temporal logic and the defender is to prevent this goal [91]. A hypergame on a graph model is formulated with the attacker's incomplete/incorrect information and the defender's information about the attackers. The authors designed algorithms to compute deceptive sure-winning strategies, where the defender can lure the attacker into encountering decoys. The optimal placement of these decoys is determined using compositional synthesis techniques, revealing that the problem of optimising decoy allocation is monotone and non-decreasing.

Hypergames on graphs have also been employed to capture the asymmetrical information with one-sided payoff misperception. Stealthy deceptive winning strategies are introduced in [92], where a deceptive strategy is stealthy when their actions are subjectively rationalisable. It was shown that the deceptive sure-winning and almost-sure-winning regions are different when a player has incomplete/incorrect information.

8.2.1.5 Cyber Deception

Hypergames are also valuable for cyber deception due to their handling of uncertainty. In the work by Ferguson and Walter [60], these situations are modeled using a sequential attack-defence game. Cyber deception games are formulated using hypergame theory, where the perception of the players is the focus and is differentiated from the true parameters of the game. The hypergame model presented within this paper is useful for online learning, as the defender attempts to infer the attacker's beliefs and apply them to future decisions. The observation of the attacker interacting with the network

can be used in order to estimate the attacker's perceived payoffs. The current game tree and estimation of the attacker's perceptions allow the defender to manipulate the game. This manipulation gradually restricts the strategies available to the attacker in subsequent moves.

Level-1 hypergames can be altered to change the preference vectors to a utility matrix [167]. This modification allows the defender to employ a cyber deceptive technique with the adversary not being aware of the existence of such techniques. The hypergame unfolds over multiple rounds, with each compromised node initiating a round to establish equilibrium strategies for the players. The defender possesses full knowledge of the adversary's moves and has a finite budget of honeypots with which to place on nodes. Honeypots make the adversary spend an excessive amount of time on that node. A node with no honeypot means the adversary takes the amount of time it takes to compromise the subsequent node. The players both play a different zero-sum matrix game, which is solved by finding the snapshot equilibrium.

A diagrammatic hypergame representation was introduced in [105] called the hypergame Perception Model (HPM) which can help model misperception and deception. It is introduced using a two-player level-3 hypergame: overall this means there are eight level-0 games, four level-1 games, two level-2 games, and the final level-3 game within the hierarchy. The analysis of HPM is similar to traditional hypergames, where the perspectives and preferences are taken into account as well as the player's deception goals. An example is shown when they model the deception during the Cuban Missile Crisis.

Deception can be harnessed to obstruct attackers' access to valuable resources. Hypergames offer a means to analyse how deception can favor the defender. Cyber conflicts can be divided into three kernels: misinterpretation, over-perception, and under-perception, which are the building blocks of modeling deception. These have been used in order to model the ErsatzPassword scheme, a security control that protects hashed passwords against brute force password-cracking [69]. The study delved into two techniques: triggering an alarm if an attacker uses a cracked password or redirecting the attacker to a honeynet. It was found that the scheme works well against both risk-tolerant and risk-averse attackers, as well as being effective in blocking/catching attackers under different levels of misperception.

8.2.1.6 Repeated Hypergames

Repeated hypergames can be used to consider the detection of an attacker attempting to manipulate a system while being undetected. Different monitoring approaches and learning schemes for the defender were considered in [8]. A static cost minimisation problem is considered which is constrained within an operating envelope. The attacker can manipulate the defender's perception and their goal is to maximise the defender's cost. However, the defender can monitor these distributions and try to catch the attacker.

The defender can monitor the distribution and sweep the system if the distributions differ but this may not be enough. Attackers could resample from the distribution making the observed values indistinguishable but still return sub-optimal results. Another

approach would be to monitor the calculated cost; however, this would allow the attacker to manipulate the defender indefinitely. Finally, the defender could monitor the true cost; this means consulting an independent source after the optimisation has been performed.

8.2.1.7 Multiplayer Games

Single-leader-multiple-follower (SLMF) Stackelberg security games can be used to describe the situation between defenders and attackers, such as in a cyber-physical system. In this game, the leader is the defender, and the followers represent the attackers. Given that wireless networks often involve numerous attackers with varying capabilities, a level-2 Stackelberg hypergame is employed to analyse the SLMF game [41]. Using Hyper Nash equilibria (HNE), both the strategic and cognitive stability of the games has been provided. Connections between Misperception Strong Stackelberg equilibrium (MSSE), Deception Strong Stackelberg equilibrium (DSSE), and HNE are also established. MSSE under certain assumptions is a HNE of the single-leader-single-follower (SLSF) game. Moreover, under additional assumptions, any DSSE is a HNE of the SLSF game. Under SLMF games an MSSE is also a HNE if all followers attack the same target, as all targets are independent in this game.

A two-player hypergame can model a scenario where each player chooses between cooperative and aggressive strategies, with differing preferences. Both players possess misperceptions about the preference order of the other player. A third party, an external observer, can be introduced in order for both players to verify their misperceptions [39]. This third party knows everything both players know, so the players can ask them questions. They could inform players that one player has a limited view and explain which actions they do not see. Consequently, this can change the outcome of the overall game, as the player with more knowledge can predict a more favourable result for themselves.

8.2.1.8 Risk Assessment and Risk Management

Dynamic occlusion occurs when autonomous vehicles are in situations with three or more vehicles. For example, consider a scenario where three cars are at a crossing. Car 1 could be checking to see if they can proceed, while Car 2 is blocking the view of a third car, posing a potential collision risk. Evaluating how much dynamic occlusion contributes to driving risk uses Dynamic Occlusion Risk (DOR), [84]. Various occlusion situations are generated and level-0 and level-1 hypergames are then used to analyse them in order to compute the DOR measure for each situation. These are solved using Nash equilibria and simulate the level-0 hypergame, which would typically result in no crash, as everyone has all the information. In order to simulate the level-1 hypergame, incomplete information has to be included. The Nash equilibria of each player is calculated and if a crash occurs the game is repeated but with an emergency braking maneuver the moment the vehicle is no longer occluded.

8.2.1.9 Adversarial Environments

In real-life situations, when an attacker realises they've been obstructed, they attempt to alter their approach. This has been modeled using an adversary environment, where player one represents an agent and player two represents the adversary. During the interaction, the adversary can deduce the agent's intention and adapt its strategy, creating a more dynamic and realistic scenario [99]. Omega-regular hypergames are used for this purpose, where omega-regular games involve a zero-sum two-player game with a controllable player and an uncontrollable player. The dynamic aspect of the game is introduced using inference, assuming the uncontrollable player has complete observation capabilities. Consequently, the controllable player has a transition system for their level-1 hypergame, which captures the changes in actions and evolving game of the uncontrollable player.

8.2.2 Possible Future Applications

8.2.2.1 Cyber-physical Attack Graphs

Section 8.2.1.4 of this paper showcases the different ways hypergames have already been used in graph-based applications with AD games and simple attack graphs. Using this knowledge and the knowledge of AND/OR Graphs or Cyber-physical attack graphs (CPAG) we feel there exists the potential to enhance the realism of cyber-graphs by incorporating models of incomplete information. While Section 8.1.2 shows a new approach to how this conversion can work, however it is still early in its progress.

AND/OR graphs can be used in cybersecurity scenarios serving as tools for solving problems and problem decomposition. In these graphs, nodes represent states or goals, with their respective successors marked as either AND or OR branches. The AND branch means all parents need to be satisfied, while the OR branch requires just one to be fulfilled. Research has delved into the integration of AND/OR graphs into the realm of cybersecurity [11], as well as their application in game theory [93]. CPAG are the class of attack graphs that effectively encompasses both cyber and physical aspects of an attacker attempting to breach a network's security [85]. The cyber nodes are used to represent AND/OR operators, cyber actions and exploits, and preconditions/privileges. Meanwhile, the physical nodes contain attacks that happen in a physical aspect e.g. 'open a locked door'.

8.2.2.2 Defence Trees

Attack-defence trees enable the formal evaluation of attack and defence scenarios by representing them in a graphical and intuitive way. Game theory has been successfully applied to attack-defence trees by developing them into stochastic two-player games [5]. This is achieved by converting each sub-tree into a game and combining the games using sequential composition.

It is conceivable that these structures can also be converted into hypergames, thereby introducing the aspect of incomplete information. Sub-games can be converted to two-

player games much like before. However, each player may possess different sub-games dependent on their perception of the game.

8.2.2.3 Multiplayer Applications

Hypergames and cybersecurity have enjoyed limited research within the area of multiple attackers and defenders (Section 8.2.1.7). However, the majority of situations within real-world cybersecurity scenarios have numerous attackers or defenders. While research on multiplayer games is prevalent in general game theory, a potential approach could involve the transformation of a normal form game into a hypergame.

8.2.2.4 Sociotechnical Cybersecurity

All cyber systems exist within a broader social and environmental context, encompassing legal frameworks, regulations, and standards. This context can be critical in determining the best courses of action that correspond to acceptable levels of risk vs potential reward. Taking into account humans in (or over) the loop in cybersecurity decision-making adds complexity and uncertainty to game scenarios. Wider organisational structures, culture, and many other factors may also play into this.

8.2.2.5 Adversarial Machine Learning

Machine Learning and Artificial Intelligence introduce entirely new cybersecurity vulnerability classes, which could be analysed in a hypergames context. Hypergame theory potentially provides a useful tool to devise strategies for defending ML-based systems through the application of appropriate technical tools and capabilities for defenders. One example is the modification of data distributions to the extent that deep learning subsequently misclassifies the altered data distribution [147] (this is commonly known as an evasion attack). Adversarial data can be generated by solving for optimal attack policies in Stackelberg games where adversaries target the misclassification performance of deep learning.

A recent survey of game theoretic approaches for adversarial machine learning [169] provides further formulations of Stackelberg game variants. These include the ‘adversary as leader’ (for instance, where an ML model is adapted based on observed attacks); and ‘learner as leader’ (for instance, a pre-trained spam filter is deployed that is then attacked). A more realistic scenario is a ‘single-leader-multi-follower’ game, where multiple different attackers exist and may try different approaches to attack an ML model. This lends itself to a Bayesian Stackelberg game [45] formulation but finding optimal strategies in such a game is NP-hard, although shortcuts exist [120]. The authors identify three important features of ML model security based on game-theoretic insights: robust attribute selection, conservative strategy (minimising risk tolerance), and using randomness effectively.

Recent work has proposed an Adversarial Strategic Game for ‘Machine Learning as a Service’ using System Features [148]. An additional, recent, example of a possible future

application is in the construction of ‘jailbreaking’ prompts for the latest generation of large language models.

8.3 Suggested Research Directions

We have observed numerous exciting current and future applications of hypergames to cybersecurity. However, there are a number of aspects of hypergame theory that need to be advanced in order to better support these and other potential applications.

8.3.1 Generalisation and Transferability

Within the domain of cybersecurity, diverse use cases come with distinct rules governing each scenario. To address this diversity, hypergames must be easily and readily generalised. This allows for proofs to be applicable across desired situations without the encumbrance of creating individual proofs for each case. This aspect is particularly relevant when considering the changeable nature of cybersecurity. Depending on the kind of cybersecurity scenario being presented, the game representation may change daily, hourly, or even more frequently during an attack. The absence of certainty over the validity of the scenario or the game as the landscape changes, means that acting on the results may not be advisable.

An example of this would be the discovery of a new zero-day vulnerability. In such cases, a general framework is imperative, one that accommodates the addition of a new attack vector that could depend on attacker properties, methods, or data previously unknown prior to its identification.

Similarly, the role of transferring the learning from one scenario to another is similarly required. In many security scenarios, the data needed to make decisions is often sparse or incomplete. In these cases, the decisions being made are not always correctly informed. A space where transfer learning can support the decision-making process is beneficial.

Specifically, where there is limited data to support the creation of a new game-theoretic model, the concept of transferability will be inherently useful. The concept behind transferability is that there are some domains of security that are better represented by data, design, or expert knowledge. This means that decisions can be made with a greater degree of confidence in some domains than others. In these cases, it would be beneficial to take the learned principles and apply them to domains that have similar properties but lack the specific data to support the development of a new model.

A potential example would be the design of a new architecture. Not all of the details of a new architecture may be fully realised as the data on usage would not be known, however, the security principles need to be tested. The challenge is that new architecture may not share the same common structures or features of a previous architecture. In the space of cyber-physical systems, the digitalisation of a system may involve new components that may have only been seen in another context, such as a different industry or in a more analogue system. In this case, being able to transfer the knowledge from the domain of one architecture to that of a different architecture without the need for creating a game from first principles, becomes invaluable.

Both the creation of general forms and the adoption of transferable learning approaches share the common goal of simplifying the process of engaging with new or evolving scenarios. Applying these principles offers a way to explore intricate scenarios with the flexibility in strategy development that comes from the hypergames foundation.

8.3.2 Adaptive Games

As highlighted in the previous section, the landscape of cybersecurity is constantly changing and evolving. The foundations of hypergames are well suited to capturing some of the nuances of these changes, such as changes in attacker priorities or the implementation of new known approaches to defence. Whilst this may support the smaller day-to-day changes of cybersecurity, it does not account for some of the bigger shifts in the way that attackers and defenders might act.

The most prominent example of this would be threat intelligence. Different threat intelligence source have sources and foci that will highlight different actors or vectors. Introducing an additional source or swapping sources can potentially alter the tools, techniques, and procedures prospective attackers might employ. This, in turn, can reshape attack pathways, available strategies, and even the rules governing the actors' interactions. Here, the game is still conceptually the same, but some of the underlying conditions or assumptions may vary.

By altering the profile of attackers within the game, the rules themselves may need to be changed to reflect this. In an active security system, the games must have the most current information to provide trustworthy results. To achieve this, further study is needed to provide an understanding of the application and limitation of adaptive rule sets in hypergames.

This aspect holds particular significance for defences that are designed to change or be non-permanent. The two most common approaches are self-adaptive defences and cyber-deception. In both domains, employing a game theoretic approach to analyse the state of the system and making real-time decisions on a given state provides an alternative to current data-driven approaches. In this case, adaptive games would allow for the development of defensive scenarios that are tailored to the current security state, offering options such as the fidelity of different honeypots as a security response. These strategic options might only be reasonable to include based on the data gathered from an active attack, such as an attacker's position in a network. Additionally, by being able to apply a mixed strategy solution to security checks or the resources allocated to a deceptive environment, there are efficiencies that can be gained from resource reduction or maintaining system usability.

8.3.3 Non-Standard Multistage Games

An area of game theory for security that can often be simplified is the level of knowledge that each player has of the state of the world. This is naturally one benefit of the hypergames approach to security games. With a constantly evolving set of available

strategies, the idea is that as more knowledge is gained the possible strategy space expands to include these new options.

One challenge that exists in this area is that the states of the game might not have any viable end goals available. For example, an attacker launching an attack against a system might be able to build a full understanding of the perimeter security of a target but have limited to no knowledge of what lies beyond and potential end targets. This means that at best the attacker has a partial mapping of the game and potentially no known end states.

There is an argument to be made that the two scenarios present different game formulations, with an attack at the perimeter being a Stackleberg game, where the defender has committed to an observable defensive strategy that the attacker can observe, whilst inside the perimeter the same knowledge and observation cannot be applied - unless considered over a prolonged period of maintained presence, which creates different challenges for representation.

To illustrate this, consider an attacker who has identified the initial stages of an attack against a defender considering the breadth of the available perimeter defences or initial routes of compromise. However, even if this initial stage of attack succeeds, there's no guarantee of a predetermined system, set of defenses, or approach that the attacker can exploit to bring the attack to completion. An entry point gives rise to the need for further investigation by the attacker to identify subsystems, which can result in additional subgames, which in themselves need to be accounted for in the design. Naturally, this requires a determined attacker that has a specific target in mind; realistic examples of this are in the realms of cyber-espionage and hacktivism. The former might have specific information they need to uncover in the system, whilst the latter might be more opportunistic in their approach to the later stages of the game.

To address this challenge, further study is needed into multistage hypergames, particularly those where there are different rules governing the game at each stage. Understanding how to perform a game where future stages of the game might not be mapped or mappable until a later time needs to be understood so as to capture more realism in attack-defence scenarios.

8.3.4 Multiplayer Games

The examples of hypergames have thus far been limited to 2-player games, but in many cases with security, there are more actors that are involved. Sometimes these actors are competitive, such as in target selection; other times they may be collaborative or neutral, as would be the case with multiple attackers.

For competitive multiplayer games, there is a consideration that different parties might have different architectures, priorities, and thus strategies, whilst still effectively acting in similar roles. Thus, having different degrees of understanding and visibility of the threat landscape can open up different approaches for defenders.

In target selection, there is a single attacker who is looking to perform an attack against one of a number of targets. A game can be constructed where the defenders are effectively competing against each other to deter the attacker from targeting them.

Within a hypergame framework, the potential defenders have to decide how to defend their vulnerable paths. However, unlike an attacker-defender 2-player game, there is no guarantee that they will be attacked, as the attacker has a multitude of targets to select from. As such the game becomes assigning the minimum amount of resources needed to either no longer be a target, or minimise the probability they are selected as a target. This would all be done with incomplete information with regard to the number of other targets and the strategies that they have.

For establishing a collaborative game, there is a requirement for a degree of non-obvious strategic withholding. This might be aligned with the target selection problem, where the overall desire is to make the landscape as difficult for an attacker as possible whilst spending the fewest resources. Thus, by working collaboratively the state of security as a whole is improved reducing the wider risk, but in the face of a determined attacker, a target will still be selected. Investing resources should therefore be done to maximise your own defence at a minimal cost, whilst exploiting resources from other parties to support their own defence.

A collaborative example can be observed in threat intelligence sharing. Here, the parties seek to share the minimum amount of information on attacks or threats with a community whilst getting the greatest possible level of intelligence from the community. A hypergame formulation can model the range of attacks that adversaries have access to and present the scope of strategies that are available for defence. The defenders only have access to a view of the attacker's strategies that are given by known threats to them directly or from the community. In this scenario, if nobody shares knowledge then an attacker will have many paths that are potentially undefended; whereas, through collaboration, there would be better available strategic coverage of these attacks. Importantly this also provides data driven insights into common co-operative game strategies such as freeloading and withholding. In both cases there is potential risk from the collective from not having better information against a wide range of attackers, where using hypergames will allow for a more detailed exploration of the impact that this causes.

By attaining a deeper understanding of the dynamics of multiplayer hypergames, it becomes feasible to provide enhanced support for assessing more intricate scenarios. This work would help support a more complete view of how and why threat information-sharing platforms are not utilised to their fullest, and build further understanding of the principle of cyber-deterrence.

8.3.5 Exploring Limitations

The preceding research areas have predominantly focused on advancing methods and approaches to formulate more intricate applied scenarios. Nevertheless, there remain unresolved questions pertaining to hypergames themselves that must be addressed to instil confidence in their utilisation.

The most prominent possible limitations concern data requirements. One of the most considerable issues with any data-driven approach to security is that of data availability and quality, to which hypergames is no exception. Many of the conventional data-

related challenges in game-theoretic models persist. Constructing the games hinges on insights into players' possible strategies, asset valuations, and the effectiveness of security measures. Typically, these values are difficult to obtain as they can be conditional or speculative. As such, there needs to be further study into the degree to which these conventional challenges are maintained or exacerbated by the added complexity of hypergames. Similarly, it is not clear the extent to which new data challenges exist that are hypergames specific.

With additional levels of interaction and strategy being made available using hypergames, there is a further question relating to where interdisciplinary research is needed to help support the application of hypergames. In particular, aspects of psychology seem to be relevant, with the notion that understanding how to develop structures of decision and reward such that the games are being played by appropriate rational actors. Likewise, taking into account economic and business-related research to better inform decisions surrounding the valuation of assets, losses, and other financial factors would improve the underlying models being acted upon. Lastly, the overlap with machine learning topics has many commonalities in design and challenges, which are discussed in the next section.

8.4 Related Research Areas

While providing an exhaustive review of modern Reinforcement Learning (RL), Multi-Agent Reinforcement Learning (MARL), and game theory exceeds the scope of this paper, it's important to note that (MA)RL constitutes a crucial research area for extending hypergame research. Indeed, while game theory enjoys major use in RL and MARL methodologies and applications, hypergame theory has been scarcely explored even in domains where it may be more appropriate than classical game theory.²

Informally, the 'value' of a state $x \in \mathcal{S}$ under a policy π represents the expected future rewards when starting in state x and following the sequence of state and actions induced by π thereafter. Given Markov Decision Process (MDP), the *state*-value function for the policy π is defined as

$$V_\pi(x) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \beta^t r(X_{t+1}, \pi(X_{t+1})) | X_t = x \right],$$

where $\beta \in [0, 1]$ is a discount factor. Analogously, we define the value of taking action $a \in \mathcal{A}$ in state $x \in \mathcal{S}$ under π as the expected future rewards from taking action a in state x and then following π :

$$Q_\pi(x, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \beta^t r(X_{t+1}, \pi(X_{t+1})) | X_t = x, A_t = a \right].$$

²We discuss the discounted reward case here to establish intuition about the connections between value functions.

Q_π is called the *action-value* function for policy π . We suppress the dependence on π for both value functions if there is no risk of confusion.

Let $\pi(a|x)$ be the probability of choosing $a \in \mathcal{A}$ in state $x \in \mathcal{S}$ under policy π . The foundational concept of optimality in RL, called Bellman Optimality, arises from a fundamental recursive relationship called the Bellman Equation ([149]).

$$\begin{aligned} V_\pi(x) &= \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \beta^t r(X_{t+1}, \pi(X_{t+1})) | X_t = x \right] \\ &= \sum_a \pi(a|x) \sum_{x', r} p(x', r|x, a) \left[r + \beta V_\pi(x') \right] \end{aligned}$$

The Bellman Equation is also called the *self-consistency* condition ([149]). For any $x \in \mathcal{S}$ and $a \in \mathcal{A}$, denote the optimal value functions by $V^*(x) = \max_\pi V_\pi(x)$ and $Q^*(x, a) = \max_\pi Q_\pi(x, a)$. V^* and Q^* are linked by the following relationship ([149]).

$$Q^*(x, a) = \mathbb{E}_{\pi^*} [r(X_{t+1}) + \beta V_*(X_{t+1}) | X_t = x, A_t = a].$$

Similarly, for V^* we have

$$V^*(x) = \max_{a \in \mathcal{A}} Q^*(x, a).$$

To see how we arrive at the Bellman Optimality equation, note that V_* is the value function for π^* , so V^* must also satisfy the Bellman Equation. Hence, the previous equation with the self-consistency condition becomes the Bellman Optimality equation,

$$\begin{aligned} V^*(x) &= \max_{a \in \mathcal{A}} Q^*(x, a) \\ &= \max_{a \in \mathcal{A}} \mathbb{E} [r(X_{t+1}) + \beta V_*(X_{t+1}) | X_t = x, A_t = a]. \end{aligned}$$

The corresponding Bellman Optimality equation for Q^* is

$$Q^*(x, a) = \mathbb{E} [r(X_{t+1}) + \beta \max_{a' \in \mathcal{A}} Q_*(X_{t+1}) | X_t = x, A_t = a].$$

Conceptually, the Bellman Optimality equation expresses that the value of a state visited by the optimal policy π^* must yield an expected return for the best action from that state ensuring optimality ([149]). Additionally, note that neither optimality equation makes a direct reference to a policy. Hence, obtaining optimal policies is straightforward if we have optimal state-value or value functions: simply choose the ‘greedy’ action, i.e., \max_a , for every state as the optimal policy. Finally, note that because we can express Q^* as an expectation, we can use stochastic approximation methods to iteratively estimate it ([151]).

In recent years, there has been notable advancement in Deep Reinforcement Learning (DRL), leveraging deep neural networks to approximate intricate value functions. To

expedite learning, efforts have concentrated on distributing the training process across multiple machines. Furthermore, Inverse Reinforcement Learning (IRL) has gained prominence. In the realm of MARL, the emphasis has been on training multiple agents to effectively interact and collaborate. This entails tackling challenges such as multi-agent credit assignment and exploration. Additionally, the incorporation of transfer learning in MARL empowers agents to extend their knowledge from one task to another.

Agent-Based Models (ABMs) delve into the behaviour and interactions of individual agents within a larger system. Lately, the link between ABMs and hypergames has gained significant attention. ABM researchers have embarked on exploring hypergames to depict intricate scenarios characterised by intricate interactions and dynamic strategies. While ABM researchers are already venturing into hypergames, an opportunity emerges for deeper integration between the domains of ABMs and MARL. Hypergames provide a structured framework for capturing the complexities of multi-layered interactions, benefiting applications in both ABMs and MARL. As interdisciplinary connections continue to strengthen, we can anticipate a richer exchange of ideas and methodologies between these evolving fields. Interestingly, one can show that ABMs are just a subset of MARL, which means hypergames should have relevance for MARL.

8.4.1 Terminology

We believe the lack of terminological overlap contributes to the gap between modern (MA)RL work, areas of game theory, and compartmental modelling (CM). For ease, we offer a table comparing analogous terms in (MA)RL, game theory (GT), evolutionary game theory (EGT), and compartmental models (CM).

(MA)RL	GT	EGT	ABMs & CMs
Environment	Game	Game	-
Agent	Player	Population	Agent
Action	Action	(Pheno)Type	Compartment/State
Policy	Strategy	Type Distribution	-
Reward	Payoff	Fitness	-

Table 8.7: Domain-Specific Terminologies.

In Table 8.7, we summarise our understanding of the terminological connection between fields. Where we have marked ‘-’ indicates that there is not a direct parallel commonly available for that category.

8.5 Conclusions

Hypergames allow situations with incomplete information to be modeled in a simplified manner. We have found that hypergames have been applied to the area of cybersecurity; however, the reach has been limited. Even if an area has research, its usually very

few papers, suggesting the large scope of research which is still available. Within this Chapter, we have looked into and suggested areas in which hypergames could be useful to expand upon as well as new areas to delve into, such as multiplayer games and cyber-physical attack graphs. As well as delving into the first steps at converting AND/OR graphs into hypergames for the purpose of cybersecurity scenarios.

Game theory and hypergame theory share common principles under favourable conditions, likely paralleling their applicability across various fields, including RL. Despite the converging trajectories of RL and other methodologies, particularly in their pursuit of incorporating computation and uncertainty quantification to address game-theoretic uncertainties, a notable gap remains. This gap, however, presents an opportunity for hypergame theory to play a complementary role within RL and other methodologies.

Part V

Conclusions

Chapter 9

Conclusions and Future Work

In a comprehensive exploration, this work illustrates the foundational principles of computable analysis and game theory. Building upon this foundation, it explored practical applications, illuminating how these theories can be harnessed to solve real-world problems.

This work has encompassed the classification of game theory techniques, shedding light on how non-computable Nash equilibria is. We were able to demonstrate that $\text{AoUC}_{[0,1]}^* \leq_W \text{Nash} \leq_W \text{AoUC}_{[0,1]}^\diamond$, we also know that $\text{AoUC}_{[0,1]}^* <_W \text{AoUC}_{[0,1]}^\diamond$, hence one of our restrictions must be strict. This is left as an open question in Section 5.4 which would be an interesting topic to research further. Moreover, if $\text{AoUC}_{[0,1]}^* <_W \text{Nash}$ then it would be interesting to explore how many players are needed to render the Weihrauch degree of finding Nash equilibria harder than the two-player case. Additionally, we explored the consequences of our classifications such as looking into Las Vegas computability and Monte Carlo computability (Section 5.5).

Moreover, we extended this computable analysis to the domain of machine learning, dissecting the diverse landscape of classifiers and learners. We presented many questions about verifying classifiers obtained by ML, starting with elementary questions before delving deeper into the nuances of topics like whether adversarial examples exist. We suggest further work in Section 7.4, regarding what classifier we would want to obtain for given training data, or whether there is a learner in the first place guaranteed to meet the criteria. Chapter 7 shows that altering conditions or considering maps with more information will generally render the maps non-computable. A follow-up to address was thought of via a special session seminar that the researcher conducted at CiE 2023 [48].

Finally, we investigated the emerging concept of hypergames in cybersecurity, looking into not only how hypergames can provide a new viewpoint to these scenarios, but also showing how we can convert typical cybersecurity frameworks into hypergames. In Section 8.5 we explain that even though hypergames are now being applied to the field of cybersecurity, the reach is limited. Therefore, most areas have not been touched or still require more research. The researcher and co-authors have started into this research by combining hypergames with AND/OR cyber attack graphs (Section 8.1.2) as well as suggesting many research directions (Sections 8.2.2 and 8.3) where hypergames would be

9. Conclusions and Future Work

a good addition such as adaptive games, defence trees and sociotechnical cybersecurity.

Bibliography

- [1] Armaghan Abed-Elmdoust, Abbas Afshar & Reza Kerachian (2011): *Developing a Waste Load Allocation Model Using General Meta Rationality and Nash Equilibriums*. *4th International Perspective on Water Resources and the Environment*.
- [2] Nathanael Ackerman, Julian Asilis, Jieqi Di, Cameron Freer & Jean-Baptiste Tristan: *On the Computable Learning of Continuous Features*. Presentation at CCA 2021.
- [3] Mostofa Ahsan, Kendall E Nygard, Rahul Gomes, Md Minhaz Chowdhury, Nafiz Rifat & Jayden F Connolly (2022): *Cybersecurity threats and their mitigation approaches using Machine Learning - A Review*. *Journal of Cybersecurity and Privacy* 2(3), pp. 527–555.
- [4] Alexander D’Amour et al (2022): *Underspecification Presents Challenges for Credibility in Modern Machine Learning*. *Journal of Machine Learning Research* 23(226), pp. 1–61. Available at <http://jmlr.org/papers/v23/20-1335.html>.
- [5] Zaruhi Aslanyan, Flemming Nielson & David Parker (2016): *Quantitative verification and synthesis of attack-defence scenarios*. In: *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, IEEE, pp. 105–119.
- [6] Ziggy Attala (2019): *AI Plays Rock-Paper-Scissors*. BSc Dissertation, Swansea University.
- [7] Craig Bakker, Arnab Bhattacharya, Samrat Chatterjee & Draguna L. Vrabie (2020): *Hypergames and Cyber-Physical Security for Control Systems*. *ACM Transactions on Cyber-Physical Systems* 4(4), pp. 1–41, doi:10.1145/3384676.
- [8] Craig Bakker, Arnab Bhattacharya, Samrat Chatterjee & Draguna L. Vrabie (2020): *Learning and information manipulation: Repeated Hypergames for cyber-physical security*. *IEEE Control Systems Letters* 4(2), pp. 295–300, doi:10.1109/lcsys.2019.2925681.
- [9] Craig Bakker, Arnab Bhattacharya, Samrat Chatterjee & Draguna L. Vrabie (2021): *Metagames and hypergames for deception-robust control*. *ACM Transactions on Cyber-Physical Systems* 5(3), pp. 1–25, doi:10.1145/3439430.

- [10] Marian A. Baroni & Douglas S. Bridges (2008): *Continuity properties of preference relations*. *Mathematical Logic Quarterly* 54(5), pp. 454–459, doi:10.1002/malq.200710059.
- [11] Cambrun M Barrere & C Hankin (2021): *Analysing mission-critical cyber-physical systems with AND/OR graphs and MaxSAT*. *ACM Transactions on Cyber-Physical Systems* 5, pp. 1–29, doi:10.1145/3451169. Available at <http://dx.doi.org/10.1145/3451169>.
- [12] Andrej Bauer (2022): *Instance reducibility and Weihrauch degrees*. *Logical Methods in Computer Science* Volume 18, Issue 3, doi:10.46298/lmcs-18(3:20)2022. Available at <http://lmcs.episciences.org/9906>.
- [13] Zach BeMent, Jeffrey Hirst & Asuka Wallace (2021): *Reverse mathematics and Weihrauch analysis motivated by finite complexity theory*. *Computability* 10(4), pp. 343–354, doi:10.3233/com-210310.
- [14] P. G. Bennett & C. S. Huxham (1982): *Hypergames and What They Do: A Soft O.R. Approach*. *The Journal of the Operational Research Society* 33(1), pp. 41–50, doi:10.2307/2581870.
- [15] Peter G. Bennett (1980): *Hypergames: Developing a model of conflict*. *Futures* 12(6), pp. 489–507, doi:10.1016/0016-3287(80)90005-1.
- [16] PG Bennett (1977): *Toward a theory of hypergames*. *Omega* 5(6), pp. 749–751, doi:10.1016/0305-0483(77)90056-1.
- [17] Lenore Blum, Felipe Cucker, Michael Shub & Steve Smale (1998): *Complexity and Real Computation*. Springer.
- [18] Vasco Brattka (2023): *On the Complexity of Computing Gödel Numbers*. Available at <https://doi.org/10.48550/arXiv.2302.04213>.
- [19] Vasco Brattka (2023): *On the Complexity of Learning Programs*. In Gianluca Della Vedova, Besik Dundua, Steffen Lempp & Florin Manea, editors: *Unity of Logic and Computation*, Springer Nature Switzerland, Cham, pp. 166–177.
- [20] Vasco Brattka, Matthew de Brecht & Arno Pauly (2012): *Closed choice and a Uniform Low Basis Theorem*. *Annals of Pure and Applied Logic* 163(8), pp. 986 – 1008, doi:<https://doi.org/10.1016/j.apal.2011.12.020>. Continuity, Computability, Constructivity: From Logic to Algorithms.
- [21] Vasco Brattka & Guido Gherardi (2011): *Effective Choice and Boundedness Principles in Computable Analysis*. *The Bulletin of Symbolic Logic* 17(1), pp. 73–117, doi:10.2178/bsl/1294186663. Available at <http://dx.doi.org/10.2178/bsl/1294186663>.

-
- [22] Vasco Brattka & Guido Gherardi (2011): *Weihrauch degrees, omniscience principles and weak computability*. *The Journal of Symbolic Logic* 76(1), pp. 143–176, doi:10.2178/jsl/1294170993. Available at <http://dx.doi.org/10.2178/jsl/1294170993>.
- [23] Vasco Brattka, Guido Gherardi & Rupert Hölzl (2015): *Probabilistic computability and choice*. *Information and Computation* 242, pp. 249–286, doi:10.1016/j.ic.2015.03.005. Available at <http://dx.doi.org/10.1016/j.ic.2015.03.005>.
- [24] Vasco Brattka, Guido Gherardi & Arno Pauly (2017): *Weihrauch Complexity in Computable Analysis*, chapter 2,7, pp. 4–8, 20–32. arXiv 1707.03202. Available at <https://doi.org/10.48550/arXiv.1707.03202>.
- [25] Vasco Brattka, Peter Hertling & Klaus Weihrauch (2008): *A tutorial on computable analysis*. In Barry Cooper, Benedikt Löwe & Andrea Sorbi, editors: *New Computational Paradigms: Changing Conceptions of What is Computable*, Springer, pp. 425–491.
- [26] Vasco Brattka, Rupert Hölzl & Rutger Kuyper (2017): *Monte Carlo Computability*. In Heribert Vollmer & Brigitte Vallée, editors: *34th Symposium on Theoretical Aspects of Computer Science (STACS 2017), Leibniz International Proceedings in Informatics (LIPIcs)* 66, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, pp. 17:1–17:14, doi:10.4230/LIPIcs.STACS.2017.17. Available at <http://drops.dagstuhl.de/opus/volltexte/2017/7016>.
- [27] Vasco Brattka, Joseph Miller, Stéphane Le Roux & Arno Pauly (2019): *Connected Choice and Brouwer’s Fixed Point Theorem*. *Journal for Mathematical Logic* 19(1), doi:10.1142/S0219061319500041.
- [28] Vasco Brattka & Arno Pauly (2010): *Computation with Advice*. *Electronic Proceedings in Theoretical Computer Science* 24. Available at <http://arxiv.org/html/1006.0551>. CCA 2010.
- [29] Vasco Brattka & Arno Pauly (2018): *On the algebraic structure of Weihrauch degrees*. *Logical Methods in Computer Science* Volume 14, Issue 4, doi:10.23638/LMCS-14(4:4)2018. Available at <https://lmcs.episciences.org/4918>.
- [30] Matthew de Brecht & Arno Pauly (2017): *Noetherian Quasi-Polish spaces*. In: *26th EACSL Annual Conference on Computer Science Logic (CSL 2017), LIPIcs* 82, pp. 16:1–16:17.
- [31] Matthew de Brecht, Arno Pauly & Matthias Schröder (2020): *Overt choice. Computability*. Available at <https://arxiv.org/abs/1902.05926>.
- [32] Douglas S. Bridges (1982): *Preference and utility : A constructive development*. *Journal of Mathematical Economics* 9(1-2), pp. 165 – 185.

- [33] Douglas S. Bridges (2004): *First steps in constructive game theory*. *Mathematical Logic Quarterly* 50(4-5), pp. 501–506, doi:10.1002/malq.200310115. Available at <https://onlinelibrary.wiley.com/doi/abs/10.1002/malq.200310115>.
- [34] Douglas S. Bridges & Fred Richman (1991): *A recursive counterexample to Debreu's theorem on the existence of a utility function*. *Mathematical Social Sciences* 21(2), pp. 179 – 182.
- [35] V. Bubelis (1979): *On equilibria in finite games*. *International Journal of Game Theory* 8(2), pp. 65–79, doi:10.1007/BF01768703. Available at <https://doi.org/10.1007/BF01768703>.
- [36] Antonin Callard & Mathieu Hoyrup (2020): *Descriptive Complexity on Non-Polish Spaces*. In Christophe Paul & Markus Bläser, editors: *37th International Symposium on Theoretical Aspects of Computer Science (STACS 2020)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 154, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 8:1–8:16, doi:10.4230/LIPIcs.STACS.2020.8. Available at <https://drops.dagstuhl.de/opus/volltexte/2020/11869>.
- [37] Nicholas Carlini & David Wagner (2017): *Towards Evaluating the Robustness of Neural Networks*. In: *IEEE Symposium on Security and Privacy*, pp. 39–57.
- [38] Douglas Cenzer & Jeffrey Remmel (1992): *Recursively presented games and strategies*. *Mathematical Social Sciences* 24(2), pp. 117 – 139, doi:[https://doi.org/10.1016/0165-4896\(92\)90059-E](https://doi.org/10.1016/0165-4896(92)90059-E). Available at <http://www.sciencedirect.com/science/article/pii/016548969290059E>.
- [39] B Chaib-Dara (2001): *Hypergame analysis in multiagent environments*. In: *AAAI Spring Symposia*, (Stanford, CA), pp. 147–149.
- [40] Xi Chen, Xiaotie Deng & Shang-Hua Teng (2009): *Settling the Complexity of Computing Two-Player Nash Equilibria*. *J. ACM* 56(3), doi:10.1145/1516512.1516516. Available at <https://doi.org/10.1145/1516512.1516516>.
- [41] Zhaoyang Cheng, Guanpu Chen & Yiguang Hong (2022): *Single-Leader-Multiple-followers Stackelberg security game with Hypergame framework*. *IEEE Transactions on Information Forensics and Security* 17, pp. 954–969, doi:10.1109/tifs.2022.3155294.
- [42] Jin-Hee Cho, Mu Zhu & Munindar Singh (2019): *Modeling and analysis of Deception Games based on Hypergame theory*. *Autonomous Cyber Deception*, pp. 49–74, doi:10.1007/978-3-030-02110-8_4.
- [43] Vittorio Cipriani, Alberto Marcone & Manlio Valenti (2022): *The Weihrauch lattice at the level of $\mathbf{\Pi}_1^1$ – CA_0 : the Cantor-Bendixson theorem*. Available at <https://doi.org/10.48550/arXiv.2210.15556>.

-
- [44] Vittorio Cipriani & Arno Pauly (2023): *The Complexity of Finding Supergraphs*. In Gianluca Della Vedova, Besik Dundua, Steffen Lempp & Florin Manea, editors: *Unity of Logic and Computation*, Springer Nature Switzerland, Cham, pp. 178–189.
- [45] Vincent Conitzer & Tuomas Sandholm (2006): *Computing the optimal strategy to commit to*. In: *Proceedings of the 7th ACM conference on Electronic commerce*, pp. 82–90.
- [46] Tonicha Crook (2019): *Game Theory, Games with Incomplete Information*. Master’s thesis, University of South Wales.
- [47] Tonicha Crook, Jay Morgan, Arno Pauly & Markus Roggenbach (2023): *A Computability Perspective on (Verified) Machine Learning*. *Recent Trends in Algebraic Development Techniques - 26th IFIP WG 1.3 International Workshop, WADT 2022*.
- [48] Tonicha Crook, Jay Paul Morgan, Arno Pauly & Markus Roggenbach (2023): *Exploring the Non-Computability of Machine Learning Classifiers*. *Unity of Logic and Computation*, pp. 26–28, doi:10.1007/978-3-031-36978-0.
- [49] Tonicha Crook & Arno Pauly (2021): *The Weihrauch degree of finding Nash equilibria in multiplayer games*. Available at <https://doi.org/10.48550/arXiv.2109.00972>.
- [50] Pedro Dal Bo & Guillaume R. Frechette (2013): *Strategy choice in the infinitely repeated prisoners’ dilemma*. *SSRN Electronic Journal*, doi:10.2139/ssrn.2292390.
- [51] Carl Mummert Damir D. Dzhafarov (2022): *Reverse Mathematics: Problems, Reductions, and Proofs*. Springer Cham, doi:10.1007/978-3-031-11367-3. Available at <https://doi.org/10.1007/978-3-031-11367-3>.
- [52] Soural Dandothi (2023): *Hypergames in Cybersecurity Implementation*. Master’s dissertation, Swansea University.
- [53] Constantinos Daskalakis, Alex Fabrikant & Christos H. Papadimitriou (2006): *The game world is flat: The complexity of Nash equilibria in succinct games*. In: *Proc. ICALP*, pp. 513–524.
- [54] Constantinos Daskalakis, Paul W. Goldberg & Christos H. Papadimitriou (2006): *The Complexity of Computing a Nash Equilibrium*. In: *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing, STOC 06*, Association for Computing Machinery, New York, NY, USA, pp. 71–78, doi:10.1145/1132516.1132527. Available at <https://doi.org/10.1145/1132516.1132527>.
- [55] Adam R. Day, Rod Downey & Linda Westrick (2022): *Three topological reducibilities for discontinuous functions*. *Transactions of the American Mathematical Society, Series B* 9(28), pp. 859–895, doi:10.1090/btran/115.

- [56] F. G. Dorais, D. D. Dzhafarov, J. L. Hirst, J. R. Mileti & P. Shafer (2016): *On uniform relationships between combinatorial problems*. *Transactions of the AMS* 368, pp. 1321–1359, doi:10.1090/tran/6465. ArXiv 1212.0157.
- [57] Damir D. Dzhafarov, Denis R. Hirschfeldt & Sarah Reitzes (2022): *Reduction games, provability and compactness*. *Journal of Mathematical Logic* 22(03), p. 2250009, doi:10.1142/S021906132250009X. Available at <https://doi.org/10.1142/S021906132250009X>.
- [58] Damir D. Dzhafarov, Reed Solomon & Keita Yokoyama (2023): *On the first-order parts of problems in the Weihrauch degrees*.
- [59] Kousha Etessami & Mihalis Yannakakis (2010): *On the Complexity of Nash Equilibria and Other Fixed Points*. *SIAM Journal on Computing* 39(6), pp. 2531–2597, doi:10.1137/080720826. Available at <https://doi.org/10.1137/080720826>.
- [60] Kimberly Ferguson-Walter, Sunny Fugate, Justin Mauger & Maxine Major (2019): *Game theory for Adaptive Defensive Cyber Deception*. *Proceedings of the 6th Annual Symposium on Hot Topics in the Science of Security - HotSoS '19*, doi:10.1145/3314058.3314063.
- [61] Niall M. Fraser & Keith W. Hipel (1979): *Solving complex conflicts*. *IEEE Transactions on Systems, Man, and Cybernetics* 9(12), pp. 805–816, doi:10.1109/tsmc.1979.4310131.
- [62] Bahman Ghahesifard & Jorge Cortes (2012): *Evolution of players' misperceptions in Hypergames under Perfect observations*. *IEEE Transactions on Automatic Control* 57(7), pp. 1627–1640, doi:10.1109/tac.2011.2180113.
- [63] Guido Gherardi & Alberto Marcone (2009): *How incomputable is the separable Hahn-Banach theorem?* *Notre Dame Journal of Formal Logic* 50(4), pp. 393–425, doi:10.1215/00294527-2009-018.
- [64] Guido Gherardi, Alberto Marcone & Arno Pauly (2019): *Projection operators in the Weihrauch lattice*. *Computability* 8(3-4), pp. 281–304, doi:10.3233/com-180207.
- [65] Jun Le Goh, Arno Pauly & Manlio Valenti (2021): *Finding descending sequences through ill-founded linear orders*. *Journal of Symbolic Logic* 86(2), doi:10.1017/jsl.2021.15.
- [66] Paul W. Goldberg & Christos H. Papadimitriou (2006): *Reducibility among equilibrium problems*. In Jon M. Kleinberg, editor: *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, ACM, pp. 61–70, doi:10.1145/1132516.1132526. Available at <https://doi.org/10.1145/1132516.1132526>.
- [67] Ian Goodfellow, Yoshua Bengio & Aaron Courville (2016): *Deep Learning*. MIT Press.

-
- [68] Tianbo Gu, Allaukik Abhishek, Hao Fu, Huanle Zhang, Debraj Basu & Prasant Mohapatra (2020): *Towards learning-automation IoT attack detection through reinforcement learning*. In: *2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, IEEE, pp. 88–97.
- [69] Christopher N. Gutierrez, Mohammed H. Almeshekeh, Saurabh Bagchi & Eugene H. Spafford (2018): *A Hypergame Analysis for ErsatzPasswords*. In: *33th IFIP International Conference on ICT Systems Security and Privacy Protection (SEC), ICT Systems Security and Privacy Protection AICT-529*, Springer International Publishing, Poznan, Poland, pp. 47–61, doi:10.1007/978-3-319-99828-2. Available at <https://hal.inria.fr/hal-02023738>.
- [70] Shaun Hargreaves-Heap & Yanis Varoufakis (1995): *Game theory: A critical introduction*. Routledge.
- [71] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt & Bernhard Scholkopf (1998): *Support vector machines*. *IEEE Intelligent Systems and their applications* 13(4), pp. 18–28.
- [72] Kojiro Higuchi & Arno Pauly (2013): *The degree structure of Weihrauch-reducibility*. *Logical Methods in Computer Science* Volume 9, Issue 2, doi:10.2168/LMCS-9(2:2)2013. Available at <https://lmcs.episciences.org/1124>.
- [73] Keith W. Hipel, Muhong Wang & Niall M. Fraser (1988): *Hypergame Analysis of the Falkland/Malvinas Conflict*. *International Studies Quarterly* 32(3), pp. 335–358, doi:10.2307/2600446. Available at <https://doi.org/10.2307/2600446>.
- [74] Denis Hirschfeldt (2014): *Slicing the Truth: On the Computability Theoretic and Reverse Mathematical Analysis of Combinatorial Principles*. World Scientific.
- [75] Denis R Hirschfeldt & Carl G Jockusch Jr (2016): *On notions of computability-theoretic reduction between Π^1_2 principles*. *Journal of Mathematical Logic* 16(01), p. 1650002.
- [76] Arne Holst (2021): *Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025*. Statista, June.
- [77] James Thomas House & George Cybenko (2010): *Hypergame theory applied to cyber attack and defense*. In: *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense IX*, 7666, International Society for Optics and Photonics, p. 766604.
- [78] Xiaowei Huang, Marta Kwiatkowska, Sen Wang & Min Wu (2017): *Safety Verification of Deep Neural Networks*. In: *International Conference on Computer Aided Verification*, pp. 3–29.
- [79] Yadigar Imamverdiyev (2014): *A hypergame model for information security*. *International Journal of Information Security Science* 3(1), pp. 148–155.

- [80] Radoslav Ivanov, Taylor J. Carpenter, James Weimer, Rajeev Alur, George J. Pappas & Insup Lee (2020): *Verifying the Safety of Autonomous Systems with Neural Network Controllers*. *ACM Trans. Embed. Comput. Syst.* 20(1), doi:10.1145/3419742.
- [81] Mats Jirstrand (1995): *Cylindrical Algebraic Decomposition - an Introduction*. Automatic Control Reports, Linköping University.
- [82] Dylan K H Jones (2019): *The Tokens and Boxes Game – Solving a game from mathematical theory in practice*. BSc Dissertation, Swansea University.
- [83] Dejan Jovanovic & Leonardo de Moura (2012): *Solving non-linear arithmetic*. *ACM Commun. Comput. Algebra* 46(3/4), pp. 104–105, doi:10.1145/2429135.2429155. Available at <https://doi.org/10.1145/2429135.2429155>.
- [84] Maximilian Kahn (2021): *Dynamic-Occlusion-Aware Risk Identification for Autonomous Vehicles Using Hypergames*. Master’s thesis, University of Waterloo, <http://hdl.handle.net/10012/17774>.
- [85] Theocharis-Alexandros Karathymios (2021): *Cyber-Physical Attack Graphs*. Master’s thesis, University of Cyprus, Faculty of Engineering.
- [86] Takayuki Kihara & Arno Pauly (2016): *Dividing by Zero – How Bad Is It, Really?* In Piotr Faliszewski, Anca Muscholl & Rolf Niedermeier, editors: *41st Int. Sym. on Mathematical Foundations of Computer Science (MFCS 2016), Leibniz International Proceedings in Informatics (LIPIcs)* 58, Schloss Dagstuhl, pp. 58:1–58:14, doi:10.4230/LIPIcs.MFCS.2016.58.
- [87] Takayuki Kihara & Arno Pauly (2019): *Convex choice, finite choice and sorting*. Available at <https://doi.org/10.48550/arXiv.1905.03190>.
- [88] Takayuki Kihara & Arno Pauly (2019): *Finite choice, convex choice and sorting*. In T V Gopal & Junzo Watada, editors: *Theory and Applications of Models of Computation, Theoretical Computer Science and General Issues* 11436, Springer, pp. 378–393, doi:10.1007/978-3-030-14812-6_23.
- [89] Vicki Knoblauch (1994): *Computable Strategies for Repeated Prisoner’s Dilemma Games and Economic Behaviour* 7(3), pp. 381–389.
- [90] Abhishek N. Kulkarni & Jie Fu (2021): *A theory of Hypergames on graphs for synthesizing dynamic cyber defense with deception*. *Game Theory and Machine Learning for Cyber Security*, pp. 97–112, doi:10.1002/9781119723950.ch6.
- [91] Abhishek N. Kulkarni, Jie Fu, Huan Luo, Charles A. Kamhoua & Nandi O. Leslie (2020): *Decoy allocation games on graphs with temporal logic objectives*. *Lecture Notes in Computer Science*, pp. 168–187, doi:10.1007/978-3-030-64793-3_9.

-
- [92] Abhishek N. Kulkarni, Huan Luo, Nandi O. Leslie, Charles A. Kamhoua & Jie Fu (2021): *Deceptive labeling: Hypergames on graphs for stealthy deception*. *IEEE Control Systems Letters* 5(3), pp. 977–982, doi:10.1109/lcsys.2020.3008078.
- [93] Vipin Kumar, Dana S Nau & Laveen N Kanal (1988): *A general branch-and-bound formulation for and/or graph and game tree search*. In: *Search in Artificial Intelligence*, Springer, pp. 91–130.
- [94] Marta Z. Kwiatkowska (2019): *Safety Verification for Deep Neural Networks with Provable Guarantees (Invited Paper)*. In Wan Fokkink & Rob van Glabbeek, editors: *30th International Conference on Concurrency Theory (CONCUR 2019), Leibniz International Proceedings in Informatics (LIPIcs)* 140, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, pp. 1:1–1:5. Available at <http://drops.dagstuhl.de/opus/volltexte/2019/10903>.
- [95] Stéphane Le Roux & Arno Pauly (2014): *Weihrauch degrees of finding equilibria in sequential games*. arXiv:1407.5587. Available at <http://arxiv.org/abs/1407.5587>.
- [96] Stéphane Le Roux & Arno Pauly (2015): *Finite choice, convex choice and finding roots*. *Logical Methods in Computer Science*, doi:10.2168/LMCS-11(4:6)2015. Available at <http://arxiv.org/abs/1302.0380>.
- [97] Stéphane Le Roux & Arno Pauly (2015): *Weihrauch Degrees of Finding Equilibria in Sequential Games*. In Arnold Beckmann, Victor Mitrană & Mariya Soskova, editors: *Evolving Computability, Lecture Notes in Computer Science* 9136, Springer, pp. 246–257, doi:10.1007/978-3-319-20028-6_25.
- [98] Klas Leino, Zifan Wang & Matt Fredrikson (2021): *Globally-Robust Neural Networks*. In: *Proceedings of the 38th International Conference on Machine Learning*, PMLR, pp. 6212–6222.
- [99] Lening Li, Haoxiang Ma, Abhishek N. Kulkarni & Jie Fu (2022): *Dynamic hypergames for synthesis of deceptive strategies with temporal logic objectives*. *IEEE Transactions on Automation Science and Engineering*, pp. 1–12, doi:10.1109/tase.2022.3150167.
- [100] Ofir Lindenbaum, Jay S. Stanley, Guy Wolf & Smita Krishnaswamy (2018): *Geometry based data generation*. In: *Advances in Neural Information Processing Systems*, pp. 1400–1411.
- [101] Changliu Liu, Tomer Arnon, Christopher Lazarus, Christopher A. Strong, Clark W. Barrett & Mykel J. Kochenderfer (2021): *Algorithms for Verifying Deep Neural Networks*. *Found. Trends Optim.* 4(3-4), pp. 244–404, doi:10.1561/24000000035.
- [102] Matthew Llewyn (2020): *Artificial Intelligence in Game Theory*. BSc Dissertation, Swansea University.

- [103] Robbie Gallucci Maddy Ell: *Cyber Security Breaches Survey 2022*. Available at <https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2022/cyber-security-breaches-survey-2022>. Accessed: 25/09/2023.
- [104] Michael Maschler, Eilon Solan & Shmuel Zamir (2013): *Game Theory*. Cambridge University Press, doi:10.1017/CBO9780511794216.
- [105] Mark E Mateski, Thomas A Mazzuchi & Shahram Sarkani (2010): *The hypergame perception model: a diagrammatic approach to modeling perception, misperception, and deception*. *Military Operations Research*, pp. 21–37.
- [106] Microsoft: *Microsoft Digital Defence Report 2022*. Available at <https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RWZlsV?culture=en-gb&country=gb>. Accessed: 25/09/2023.
- [107] Bhubaneswar Mishra (1993): *Algorithmic algebra*. Springer.
- [108] Jay Morgan, Adeline Paiement, Arno Pauly & Monika Seisenberger (2021): *Adaptive Neighbourhoods for the Discovery of Adversarial Examples*. arXiv preprint arXiv:2101.09108.
- [109] John H. Nachbar & William R. Zane (1996): *Non-computable strategies and discounted repeated games*. *Economic Theory* 8, pp. 103–122.
- [110] John F. Nash (1950): *Equilibrium Points in N-Person Games*. *Proceedings of the National Academy of Sciences* 36(1), pp. 48–49. Available at <http://www.pnas.org/content/36/1/48.short>.
- [111] Ali Bou Nassif, Manar Abu Talib, Qassim Nasir & Fatima Mohamad Dakalbab (2021): *Machine learning for anomaly detection: A systematic review*. *Ieee Access* 9, pp. 78658–78700.
- [112] John Von Neuman & Oskar Morgenstern (1944): *Theory of games and economic behavior*. Princeton University.
- [113] Eike Neumann (2021): *Decision problems for linear recurrences involving arbitrary real numbers*. *Logical Methods in Computer Science*. Available at <https://arxiv.org/abs/2008.00583>.
- [114] Hugo Nobrega & Arno Pauly (2019): *Game characterizations and lower cones in the Weihrauch degrees*. *Logical Methods in Computer Science* 15(3), pp. 11.1–11.29, doi:10.23638/LMCS-15(3:11)2019.
- [115] Frans Oliehoek (2005): *Game theory and AI: a unified approach to poker games*. Master of artificial intelligence dissertation, University of Amsterdam.

-
- [116] V.P. Orevkov (1963): *A constructive mapping of a square onto itself displacing every constructive point*. Soviet Mathematics IV. Translation of Doklady Akademii Nauk SSSR. Publ. by the Am. Math. Soc.
- [117] M.J. Osborne (2004): *An Introduction to Game Theory*. Oxford University Press. Available at <https://books.google.co.uk/books?id=aux1oAEACAAJ>.
- [118] Christos H. Papadimitriou (2007): *The complexity of finding Nash equilibria*. In Noam Nisan, Tim Roughgarden, Éva Tardos & Vijay Vazirani, editors: *Algorithmic Game Theory*, Cambridge University Press, pp. 29–52.
- [119] Christos H. Papadimitriou & Tim Roughgarden (2008): *Computing correlated equilibria in multi-player games*. *J. ACM* 55(3), pp. 14:1–14:29, doi:10.1145/1379759.1379762. Available at <https://doi.org/10.1145/1379759.1379762>.
- [120] Praveen Paruchuri, Jonathan P Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez & Sarit Kraus (2008): *Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games*. In: *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems—Volume 2*, pp. 895–902.
- [121] Arno Pauly (2009): *Representing Measurement Results*. *Journal of Universal Computer Science* 15(6), pp. 1280–1300.
- [122] Arno Pauly (2010): *How Incomputable is Finding Nash Equilibria?* *J. UCS* 16(18), pp. 2686–2710, doi:10.3217/jucs-016-18-2686. Available at <https://doi.org/10.3217/jucs-016-18-2686>.
- [123] Arno Pauly (2010): *Nash Equilibria and Fixed Points in the BSS-Model*. Preprint-Reihe Mathematik 6, Ernst-Moritz-Arndt-Universität Greifswald.
- [124] Arno Pauly (2012): *Computable Metamathematics and its Application to Game Theory*. Ph.D. thesis, University of Cambridge.
- [125] Arno Pauly (2015): *Many-one reductions and the category of multivalued functions*. *Mathematical Structures in Computer Science* 27(3), pp. 376–404, doi:10.1017/s0960129515000262. Available at <http://dx.doi.org/10.1017/S0960129515000262>.
- [126] Arno Pauly (2016): *On the topological aspects of the theory of represented spaces*. *Computability* 5(2), pp. 159–180, doi:10.3233/COM-150049.
- [127] Arno Pauly (2016): *On the topological aspects of the theory of represented spaces*. *Computability* 5(2), pp. 159–180, doi:10.3233/COM-150049. Available at <https://doi.org/10.3233/COM-150049>.

- [128] Arno Pauly (2019): *Effective local compactness and the hyperspace of located sets*. arXiv preprint arXiv:1903.05490.
- [129] Arno Pauly (2020): *An update on Weihrauch complexity, and some open questions*. Available at <https://doi.org/10.48550/arXiv.2008.11168>.
- [130] Arno Pauly, Cécilia Pradic & Giovanni Solda (2023): *On the Weihrauch degree of the additive Ramsey theorem*. Available at <https://doi.org/10.48550/arXiv.2301.02833>.
- [131] Arno Pauly & Florian Steinberg (2018): *Comparing Representations for Function Spaces in Computable Analysis*. *Theory of Computing Systems* 62(3), pp. 557–582, doi:10.1007/s00224-016-9745-6. Available at <https://doi.org/10.1007/s00224-016-9745-6>.
- [132] Luca Pulina & Armando Tacchella (2010): *An abstraction-refinement approach to verification of artificial neural networks*. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6174 LNCS, pp. 243–257, doi:10.1007/978-3-642-14295-6_24.
- [133] Michael O. Rabin (1958): *Effective Computability of Winning Strategies*. *Journal of Symbolic Logic* 23(2), pp. 224–224, doi:10.2307/2964422.
- [134] Stéphane Le Roux & Arno Pauly (2015): *Finite choice, convex choice and finding roots*. *Logical Methods in Computer Science* Volume 11, Issue 4, doi:10.2168/LMCS-11(4:6)2015. Available at <https://lmcs.episciences.org/1607>.
- [135] Wenjie Ruan, Xiaowei Huang & Marta Kwiatkowska (2018): *Reachability Analysis of Deep Neural Networks with Provable Guarantees*. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pp. 2651–2659.
- [136] Yasuo Sasaki (2014): *Subjective rationalizability in Hypergames*. *Advances in Decision Sciences* 2014, pp. 1–7, doi:10.1155/2014/263615.
- [137] Yasuo Sasaki & Kyoichi Kijima (2008): *Preservation of Misperceptions - Stability Analysis of Hypergames*. *Proceedings of the 52nd Annual Meeting of the ISSS - 2008, Madison, Wisconsin* 3(1). Available at <https://journals.issn.org/index.php/proceedings52nd/article/view/1007>.
- [138] Yasuo Sasaki & Kyoichi Kijima (2010): *Hypergame modeling of systems intelligent agents*. In: *2010 7th International Conference on Service Systems and Service Management*, pp. 1–6, doi:10.1109/ICSSSM.2010.5530274.
- [139] Yasuo Sasaki & Kyoichi Kijima (2012): *Hypergames and bayesian games: A theoretical comparison of the models of games with incomplete information*. *Journal of Systems Science and Complexity* 25(4), pp. 720–735, doi:10.1007/s11424-012-1074-5.

-
- [140] R.E Schapire (2003): *The Boosting Approach to Machine Learning: An Overview*, pp. 149–171. *Lecture Notes in Statistics* 171, Springer.
- [141] Matthias Schröder (2004): *Spaces allowing Type-2 Complexity Theory revisited*. *Mathematical Logic Quarterly* 50(4/5), pp. 443–459.
- [142] Shai Shalev-Shwartz & Shai Ben-David (2014): *Understanding machine learning: From theory to algorithms*. Cambridge university press.
- [143] S.G. Simpson (1999): *Subsystems of second order arithmetic*. Springer, Berlin.
- [144] Vasileios Skliris, Michael R. K. Norman & Patrick J. Sutton (2020): *Real-Time Detection of Unmodelled Gravitational-Wave Transients Using Convolutional Neural Networks*, doi:10.48550/ARXIV.2009.14611.
- [145] Robert I. Soare (2016): *Gale-Stewart Games*, pp. 217–219. Springer Berlin Heidelberg, Berlin, Heidelberg, doi:10.1007/978-3-642-31933-4_15. Available at https://doi.org/10.1007/978-3-642-31933-4_15.
- [146] E. Specker (1990): *The Fundamental Theorem of Algebra in Recursive Analysis*, pp. 264–272. Birkhäuser Basel, Basel, doi:10.1007/978-3-0348-9259-9_22. Available at https://doi.org/10.1007/978-3-0348-9259-9_22.
- [147] Aneesh Sreevallabh Chivukula, Xinghao Yang, Bo Liu, Wei Liu & Wanlei Zhou (2023): *Game Theoretical Adversarial Deep Learning*, pp. 73–149. Springer International Publishing, Cham, doi:10.1007/978-3-030-99772-4_4. Available at https://doi.org/10.1007/978-3-030-99772-4_4.
- [148] Guoxin Sun, Tansu Alpcan, Seyit Camtepe, Andrew C Cullen & Benjamin IP Rubinstein (2023): *An Adversarial Strategic Game for Machine Learning as a Service using System Features*. In: *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pp. 2508–2510.
- [149] Richard S Sutton & Andrew G Barto (2018): *Reinforcement Learning*, 2 edition. Adaptive Computation and Machine Learning series, Bradford Books, Cambridge, MA.
- [150] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow & Rob Fergus (2014): *Intriguing properties of neural networks*. *2nd International Conference on Learning Representations, ICLR 2014*.
- [151] Csaba Szepesvári (2010): *Algorithms for reinforcement learning*. Synthesis lectures on artificial intelligence and machine learning, Springer International Publishing, Cham.
- [152] S. Takahashi, N. Hinago, T. Inohara & B. Nakano (1999): *Evolutionary approach to three-person hypergame situation*. In: *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028)*, 4, pp. 254–259, doi:10.1109/ICSMC.1999.812409.

- [153] Moshe Tennenholtz (2002): *Game Theory and Artificial Intelligence*. In Mark d’Inverno, Michael Luck, Michael Fisher & Chris Preist, editors: *Foundations and Applications of Multi-Agent Systems, UKMAS Workshop 1996-2000, Selected Papers, Lecture Notes in Computer Science 2403*, Springer, pp. 49–58, doi:10.1007/3-540-45634-1_4. Available at https://doi.org/10.1007/3-540-45634-1_4.
- [154] Alan Turing (1936): *On computable numbers, with an application to the Entscheidungsproblem*. *Proceedings of the LMS* 2(42), pp. 230–265.
- [155] Alan Turing (1937): *On computable numbers, with an application to the Entscheidungsproblem: Corrections*. *Proceedings of the LMS* 2(43), pp. 544–546.
- [156] Karl Tuyls & Ann Nowé (2005): *Evolutionary game theory and multi-agent reinforcement learning*. *Knowledge Eng. Review* 20(1), pp. 63–90, doi:10.1017/S026988890500041X. Available at <https://doi.org/10.1017/S026988890500041X>.
- [157] Karl Tuyls & Simon Parsons (2007): *What evolutionary game theory tells us about multiagent learning*. *Artif. Intell.* 171(7), pp. 406–416, doi:10.1016/j.artint.2007.01.004. Available at <https://doi.org/10.1016/j.artint.2007.01.004>.
- [158] Russell R Vane (2000): *Hypergame theory for DTGT agents*. In: *American Association for Artificial Intelligence*, pp. 163–165.
- [159] K. Vela Velupillai (2009): *Uncomputability and undecidability in economic theory*. *Applied Mathematics and Computation* 215(4), pp. 1404 – 1416, doi:http://dx.doi.org/10.1016/j.amc.2009.04.051. Available at <http://www.sciencedirect.com/science/article/pii/S0096300309004172>.
- [160] K. Vela Velupillai (2011): *Towards an algorithmic revolution in Economic Theory*. *Journal of Economic Surveys* 25(3), pp. 401–430, doi:10.1111/j.1467-6419.2011.00684.x. Available at <http://dx.doi.org/10.1111/j.1467-6419.2011.00684.x>.
- [161] Muhong Wang, Keith W. Hipel & Niall M. Fraser (1989): *Solution concepts in hypergames*. *Applied Mathematics and Computation* 34(3), pp. 147–171, doi:10.1016/0096-3003(89)90102-1.
- [162] Klaus Weihrauch (1987): *Computability*. Monographs on Theoretical Computer Science, Springer-Verlag.
- [163] Klaus Weihrauch (1992): *The TTE-interpretation of three hierarchies of omniscience principles*. Informatik Berichte 130, FernUniversität Hagen, Hagen.
- [164] Klaus Weihrauch (2000): *Computable Analysis*. Springer-Verlag.

-
- [165] Linda Westrick (2021): *A note on the diamond operator*. *Computability* 10(2), pp. 107–110, doi:10.3233/COM-200295.
- [166] Matthew Wicker, Xiaowei Huang & Marta Kwiatkowska (2018): *Feature-guided black-box safety testing of deep neural networks*. In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer, pp. 408–426.
- [167] Bowei Xi & Charles A. Kamhoua (2020): *A Hypergame-Based Defense Strategy Toward Cyber Deception in Internet of Battlefield Things (IoBT)*, chapter 3, pp. 59–77. John Wiley & Sons, Ltd, doi:<https://doi.org/10.1002/9781119593386.ch3>. Available at <https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119593386.ch3>.
- [168] Jin Zhang & Jingyue Li (2020): *Testing and verification of neural-network-based safety-critical control software: A systematic literature review*. *Information and Software Technology* 123, p. 106296. Available at <https://www.sciencedirect.com/science/article/pii/S0950584920300471>.
- [169] Yan Zhou, Murat Kantarcioglu & Bowei Xi (2018): *A survey of game theoretic approach for adversarial machine learning*. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 9(3), doi:10.1002/widm.1259.
- [170] Martin Zinkevich, Michael Johanson, Michael Bowling & Carmelo Piccione (2007): *Regret Minimization in Games with Incomplete Information*. In J. Platt, D. Koller, Y. Singer & S. Roweis, editors: *Advances in Neural Information Processing Systems*, 20, Curran Associates, Inc., pp. 1729–1736. Available at https://proceedings.neurips.cc/paper_files/paper/2007/file/08d98638c6fcd194a4b1e6992063e944-Paper.pdf.

Appendix A

Robust/Sparse Code - Python

The following code was used in order to help the comprehension of the mathematical ideas robustness/sparsity and density.

```
###Libraries needed:
import matplotlib.pyplot as plt
import numpy as np
import random
from sklearn import svm
from itertools import chain

def createData():
    ### Create two random data sets of 5 points, in the range of 0-20
    x1 = random.sample(range(0, 20), 5)
    x2 = random.sample(range(0, 20), 5)
    y1 = random.sample(range(0, 20), 5)
    y2 = random.sample(range(0, 20), 5)
    data1 = zip(x1,y1)
    data2 = zip(x2,y2)
    return list(data1), list(data2)

def ModelPlot(data1,data2):
    ### Use an SVM to model the data and find the best separating line
    between the two classes and plot it
    X = np.array(data1 + data2)
    Y = np.array([0] * len(data1) + [1] * len(data2))
    clf = svm.SVC(kernel='linear', C=10000) # C is important here
    clf.fit(X, Y)
    plt.figure(figsize=(4, 4))
    # get the separating hyperplane
    w = clf.coef_[0]
    a = -w[0] / w[1]
    xx = np.linspace(0, 100)
    yy = a * xx - (clf.intercept_[0]) / w[1]
    plt.figure(1, figsize=(4, 3))
```

```

plt.clf()
plt.plot(xx, yy, "k-") #***** This is the separator line *****
plt.scatter(X[:, 0], X[:, 1], c=Y, zorder=10,
            cmap=plt.cm.Paired,edgecolors="k")
plt.xlim((0, 50))
plt.ylim((0, 50))
return xx, yy

def Model(data1,data2):
    ### Use an SVM to model the data and find the best separating line
    between the two classes
    X = np.array(data1 + data2)
    Y = np.array([0] * len(data1) + [1] * len(data2))
    clf = svm.SVC(kernel='linear', C=10000)
    clf.fit(X, Y)
    w = clf.coef_[0]
    #error checking if no separating line can be made
    if w[1] == 0:
        return [],[]
    else:
        a = -w[0] / w[1]
        xx = np.linspace(0, 100)
        yy = a * xx - (clf.intercept_[0]) / w[1]
        return xx, yy

def LeftRightLine(xx,yy):
    ### Function to check if each point is left, right or on the separating
    line
    LeftRight = [[0 for x in range(50)] for y in range(50)]
    x1= xx[0]
    x2= xx[-1]
    y1= yy[0]
    y2= yy[-1]
    for i in range(0,50):
        for j in range(0,50):
            d=(i-x1)*(y2-y1)-(j-y1)*(x2-x1)
            if d < 0:
                LeftRight[i][j] = -1 #right of line
            elif d > 0:
                LeftRight[i][j] = 1 #left of line
            else:
                LeftRight[i][j] = 0 #on the line - would be bottom
    return LeftRight

def Robust(data1,data2):
    #Create an array to store if a point is non-robust points 1, start with
    all points being robust points 0
    Robust = [[0 for x in range(50)] for y in range(50)]
    #Model original data with separating line

```

```

xx,yy = Model(data1,data2)
#Create an array for if each point is left, right or on the separating
line
LeftRight1 = LeftRightLine(xx,yy)
for i in range(0,50):
    for j in range(0,50):
        #Add one new data point
        data1.append([i,j])
        #Model new data
        xx,yy = Model(data1,data2)
        if len(xx)!=0: #Removal of errors
            LeftRight2 = LeftRightLine(xx,yy)
            for x in range(0,50):
                for y in range(0,50):
                    #Check if the point has switched sides in the new data,
                    and mark it in robust array
                    if LeftRight1[x][y] == 1 and LeftRight2[x][y] == -1:
                        Robust[x][y] = 1
                    elif LeftRight1[x][y] == -1 and LeftRight2[x][y] == 1:
                        Robust[x][y] = 1
            data1.pop(5)
    return Robust

def Sparse(data1,data2):
    #Create an array to store if a point is Sparse 1, start with all points
    being dense 0
    Sparse = [[0 for x in range(50)] for y in range(50)]
    #Model original data with separating line
    xx,yy = Model(data1,data2)
    #Create an array for if each point is left, right or on the separating
    line
    LeftRight1 = LeftRightLine(xx,yy)
    for i in range(0,50):
        for j in range(0,50):
            #Add one new data point
            data1.append([i,j])
            #Model new data
            xx,yy = Model(data1,data2)
            if len(xx)!=0: #Removal of errors
                LeftRight2 = LeftRightLine(xx,yy)
                for x in chain(range(0,i-5), range(i + 5, 50)): #Excluding a
                    square around the new data
                    for y in chain(range(0,j-5), range(j + 5, 50)):
                        #Check if the point has switched sides in the new data,
                        and mark it in Sparse array
                        if LeftRight1[x][y] == 1 and LeftRight2[x][y] == -1:
                            Sparse[x][y] = 1
                        elif LeftRight1[x][y] == -1 and LeftRight2[x][y] == 1:
                            Sparse[x][y] = 1

```

```
        data1.pop(5)
    return Sparse

def ColourGrid(data):
    color_map = {0: np.array([255, 0, 0]), # red
                1: np.array([0, 0, 255])} # blue
    # Transpose data so the grid reads teh axes in correctly
    data = np.array(data)
    data = data.T
    # make a 3d numpy array that has a color channel dimension to be sure the
    # values are always the same colour.
    data_3d = np.ndarray(shape=(50, 50, 3), dtype=int)
    for i in range(0, 50):
        for j in range(0, 50):
            data_3d[i][j] = color_map[data[i][j]]
    #plot and invert for comparisons sake
    plt.imshow(data_3d)
    ax = plt.gca()
    ax.invert_yaxis()

### Example
data1,data2 = createData()
xx,yy = ModelPlot(data1, data2) #Plot original data
robust = Robust(data1,data2)
ColourGrid(robust)
sparse = Sparse(data1,data2)
ColourGrid(sparse)
```


Appendix B

Hypergames and Hyper Nash Equilibrium - Python

The following code extends the nashpy library to also generate Hypergames, as well as the solution concepts of Hyper Nash equilibria and Stable Hyper Nash equilibria.

```
import nashpy as nash
import numpy as np
import itertools

def NashStrat(NashEq):
    #gives nash equilibria as [row number,column number]
    # returns Zeros if there is no Nash equilibria
    NGs=[]
    N=len(NashEq)
    first_elements, second_elements = zip(*NashEq)
    for n in range(N):
        icount , jcount = 0 , 0
        iStrat, jStrat = 0 , 0
        for i in first_elements[n]:
            icount += 1
            if i > 0.5: # edit this for likeliness with mixed strategies
                iStrat = icount
        for j in second_elements[n]:
            jcount += 1
            if j > 0.5: # edit this for likeliness with mixed strategies
                jStrat = jcount
        NG = [iStrat,jStrat]
        NGs.append(NG)
    return(NGs)

def HyperNash(Nash1,Nash2):
    # Finds the hyper nash equilibria for the hypergame
    Game1 = NashStrat(Nash1)
    Game2 = NashStrat(Nash2)
```

```
    ifirst_elements, isecnd_elements = zip(*Game1)
    jfirst_elements, jsecnd_elements = zip(*Game2)
    HyperNash = []
    for element in itertools.product(ifirst_elements, jsecnd_elements):
        HyperNash.append(element)
    return(HyperNash)

def StableHyperNash(Nash1, Nash2):
    Game1 = NashStrat(Nash1)
    Game2 = NashStrat(Nash2)
    Hyper = HyperNash(Nash1, Nash2)
    N = len(Hyper)
    Stable = []
    for n in range(N):
        if list(Hyper[n]) in Game1:
            if list(Hyper[n]) in Game2:
                Stable.append(Hyper[n])
    return(Stable)

## Fall of France Example
# Allies Game
Germans1=np.array([[1,2],[4,3]])
Allies1=np.array([[4,3],[1,2]])
AlliesGame=nash.Game(Germans1,Allies1)

# Germans Game
Germans2=np.array([[1,2,2],[4,3,3],[3,5,2]])
Allies2=np.array([[4,3,3],[1,2,2],[2,0,3]])
GermansGame=nash.Game(Germans2,Allies2)

# If you already know the Nash equilibria - Check
StratA1=np.array([0,1,0])
StratB1=np.array([0,0,1])
GermansGame.is_best_response(StratA1,StratB1)

StratA2=np.array([1,0])
StratB2=np.array([1,0])
AlliesGame.is_best_response(StratA2,StratB2)

# Otherwise find the Nash equilibria
NashAllies=AlliesGame.support_enumeration()
NashAllies = list(NashAllies)

NashGermans=GermansGame.support_enumeration()
NashGermans = list(NashGermans)

# Hyper Nash and Stable Hyper Nash
GameG = NashStrat(NashGermans)
GameA = NashStrat(NashAllies)
```

HyperNash(NashGermans,NashAllies)
StableHyperNash(NashGermans,NashAllies)