Quantitative Rational Verification

Thomas Steeples
Keble College
University of Oxford

A thesis submitted for the degree of
DPhil in Autonomous Intelligent Machines and Systems

Abstract

Rational verification asks whether certain temporal properties hold within the equilibria of multi-agent systems. Typically, the literature considers games where the agents traverse some concurrent game structure (or some succinct representation of it), with player goals and specifications given by LTL formulae. Whilst this is an effective framework for reasoning about a variety of different systems, it has two clear shortcomings — it assumes that agents are unrestricted in choosing their actions, and it only allows two possible player preferences over outcomes: a satisfied temporal goal and an unsatisfied one. Whilst this is sufficient for verifying binary properties (such as safety, fairness and liveness) of a system where agents can act arbitrarily, this framework lacks the ability to differentiate between non-functional behaviours of resource-constrained systems.

For an agent in a system, there may be multiple ways of having it achieve its goal, but these may have different resource requirements (such as energy usage or time expended). As such, it is natural to ask questions like 'does this fleet of autonomous cars have enough power to safely get from point A to point B?' or 'how can these industrial robots perform their respective functions, whilst collectively expending the least amount of battery power?'. These are real world problems which also invoke rich questions in theoretical computer science.

In this thesis, we introduce quantitative aspects to the framework of rational verification. We build on the existing literature by extending the traditional temporal goals of players with mean-payoff rewards; we constrain our agents by making their available actions contingent on resources they possess; and we will look at how quantitative aspects can introduce non-determinism to the rational verification setting.

As such, this will give us games with a greater strategic interest than previously analysed, bringing the theory closer to understanding real-world systems.

Contents

Preface

Authorship

For this preface and Chapter 1, I will write colloquially, and largely (but not exclusively) in the first-person singular. For Chapters 2 through 5, I will write more formally, and use the first-person plural 'we', which I like to interpret as 'you (the reader) and I'. Finally, I will again adopt a more informal tone for Chapter 6.

Material from Chapter 1 has been adapted from material originally presented as part of my Transfer of Status report. Chapter 3 is largely based off two of my papers [105, 196]. Both Chapters 4 and 5 are (as of time of writing) unpublished work, although Chapter 4 has benefited from the comments of peer reviewers. Chapters 2 and 6 contain material from all of the above sources, along with entirely new material.

I use the ACM in-text citation style throughout, which consists of giving a numbered, bracketed reference for indirect citations (*e.g.* 'SAT is **NP**-complete [76]') and the author name(s), followed by a bracketed year, for direct citations (*e.g.* 'Cook [1971] showed that SAT is **NP**-complete'). All citations are forward and backward hyperlinked to make document traversal easier, and are highlighted in a lime-green font as a visual aid.

Finally, there are certain results which appear which are essential for the presentation, but were written by one of my co-authors in the original paper. I will clearly mark such results: all material presented is mine unless otherwise stated. Additionally, there are words, sentences and paragraphs that have been tweaked as a result of comments from

my co-authors and peer-reviewers: however, I believe I have made the effort to make sure that my voice and intent have always been preserved.

Structure

Hopefully the table of contents is fairly self-explanatory, but for the sake of explicitness, let me signpost the road ahead. In Chapter 1 (Introduction), I motivate why quantitative rational verification is an important topic to study: I start by giving an informal account of why when you set out to design multi-agent systems, you inevitably end up within the rational verification framework, before reaching the same conclusion by surveying the existing landscape of literature. In Chapter 2 (Preliminaries), I help you pack your bag for the journey by walking through some material to get us all on the same page: that is, some necessary background information. Then in Chapters 3 (Themes on mean-payoff games), 4 (Iterated allocation games), and 5 (Voting games), we venture on into uncharted territory. Rather than spoil the surprise of the vistas on show in these chapters, I will just note that this is where my original contribution lies, and each chapter focuses on a different quantitative 'twist' on the basic rational verification framework. Finally, in Chapter 6 (Conclusion), I summarise my results and offer some reflection on the journey. In the back matter, I have provided a comprehensive glossary, featuring definitions of acronyms, complexity classes, and notations used, followed, of course, by a bibliography.

Intended audience

In the most literal sense, the intended audience for my thesis is simply my two examiners. However, I also hope that others may someday derive some benefit from this work. With this in mind, I note that whilst I have made every attempt to be as explicit as possible throughout, inevitably, there are a few prerequisites for reading this document. Whilst no background knowledge of game theory or model checking is assumed, it would be helpful

if you (the reader) have a basic understanding of computational complexity theory. By this, I will assume you know what a decision problem is, and what it means for a decision problem to be a member of/hard for/complete for key complexity classes such as **NP**, or **2EXPTIME**.

Acknowledgements

First and foremost, I would like to thank my supervisors, Julian Gutierrez and Michael Wooldridge, for their constant support and guidance, as well as my examiners, Jean-François Raskin and Alessandro Abate, for their invaluable feedback and comments on this thesis. Second, I would like to thank my family, Jill, Nick, and Ellie, along with my girlfriend Esther, for their endless amounts of love and patience. Third, I would like to thank Wendy Poole, for being my ever-faithful copilot.

Finally, I would like to formally acknowledge my funding bodies: I gratefully acknowledge the support of the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines and Systems EP/L015897/1, along with the Ian Palmer Memorial Scholarship.

Chapter 1

Introduction

1.1 Motivation

The bad news is that you are a terrible coder. I am sorry to be the one to have to tell you this, but it is true. The good news is that you are not the only one. *Everyone* is terrible at writing code. And this is a problem — the software industry is a hundred of billion of dollars enterprise [77], and whilst the software testing paradigm has had some success, it is fundamentally flawed, and can only ever catch a fraction of the possible errors within a program.

Additionally, the fallout from uncaught software errors can range from the mildly inconvenient to the catastrophic. Whilst we may tolerate user interface (UI) defects in an iPhone app, we certainly should not tolerate major financial loss to institutions, injury to humans, or loss of life caused by software. And yet, despite the supposed maturity of the industry, there are examples each and every year of software doing exactly this. Let us look at some famous examples of software failing unacceptably:

- In the 1980s, the Therac-25 radiation therapy machines were responsible for the deaths of at least six people [144]. Because of errors induced by incorrect concurrent programming, patients were given doses of radiation that were hundreds of

times greater than they should have been.

- In 1996, the Ariane 5 launch vehicle self-destructed 37 seconds after take-off. The cause was an integer overflow not being handled correctly. This failure cost around $370 million [83].

- In 2003, a large portion of North America lost power for up to two days, affecting an estimated 55 million people. The outage is widely accepted to have been caused by a bug in GE Energy's XA/21 system [179].

- From 2009, Toyota began to recall several models of their cars, due to the phenomenon of 'sudden unintended acceleration'. After years of recalls and legal cases, it was established that Toyota did not program safety-critical software correctly. Moreover, it was claimed by an expert witness that a single bit flip (caused by a hardware failure, or cosmic radiation) could cause the acceleration of a vehicle [130].

- In 2014, the Heartbleed vulnerability was discovered within the widely used OpenSSL library [199]. This vulnerability completely undermined the usual protection offered by SSL/TLS, implying that users' encrypted data could easily be read and understood by attackers. The vulnerability had been present in the software for over two years before it was found and fixed.

I could go on. The truth is that software failures plague virtually every industry, costing organisations millions of dollars each year, and threatening dire consequences. Traditional software testing has failed to provide the assurance that we demand from safety-critical, always-live systems.

There is an answer to this. That answer is *Formal Verification*. Formal verification is arguably one of the most fundamental activities within computer science. Within this, the main question of interest is 'Given a system and a property, does this property hold

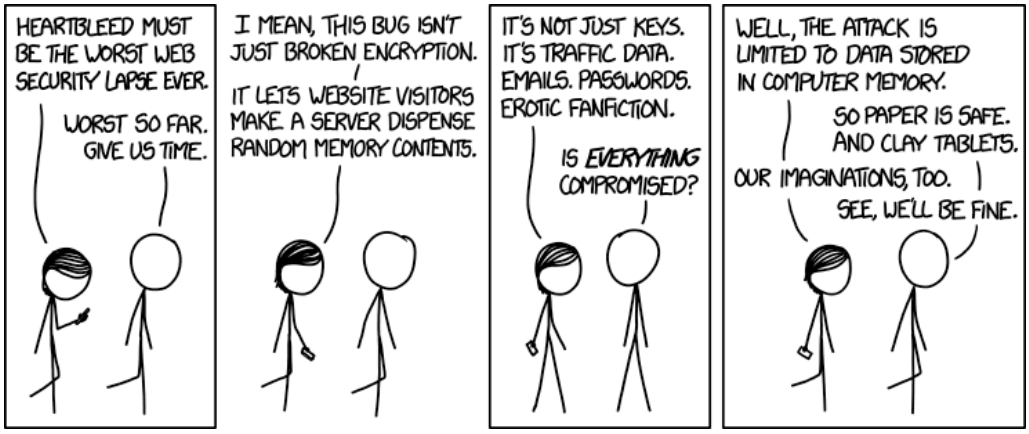

Figure 1.1: The webcomic xkcd's take on the Heartbleed bug.

in the system?'[1]. In the simplest case of software testing, you check that given a set of preconditions, after running a portion of the code, you end up in some expected set of postconditions. A stronger alternative is to define a set of postconditions, and show that *no matter what is input*, you end up in that set. This provides greater assurance that your code behaves in the way you expect it to.

Whilst formal verification is used in certain fields such as hardware design [116] and avionics [195], and whilst there is actually a rich wealth of tooling available (there are too many to list without doing a disservice to someone, but as a selection, there are now mature model checkers [73, 75, 120], theorem provers [39, 160, 168, 194], SMT solvers [37, 159], and of course, every man, woman and child has developed their own SAT solver these days [123]), it certainly does not have widespread adoption. Thus, given the stakes, it is crucial that *a*) industry more widely accepts formal verification as an additional tool in software development and *b*) academia provides the theory and techniques to support this wider acceptance.

There are a number of ways of formalising formal verification, but the approach we are most interested in is that of *model checking*. Broadly speaking, model checking encodes the

[1]We use the word 'system' in a very general, almost layman's sense here — this is appropriate as formal verification can be applied in a very wide variety of different scenarios. We will explore in further detail what these systems might look like momentarily.

system in question as some abstract, finite model, places some logical structure on top of it, then asks whether a certain logical formula holds. How exactly that model is encoded, and what logic we use to ask questions about it, is up for grabs. However, alarm bells should immediately ring here. Why? Because this problem, in its most general form, is undecidable [182, 201]. As such, research has focussed on exploring representations which are tractable, yet expressive enough to model systems of interest. We will explore these representations momentarily.

However, model checking is not the end of the discussion. Suppose you are a software developer and you have decided to build your answer to the iPhone — the weDial©. You build all the constituent components of your new phone: the baseband radio, the user interface — you even write your own kernel. You conduct unit testing on each component and they all work splendidly; you declare your product a success, ship it to millions and before you know it, the negative reviews start rolling in: the phone does not work! The radio interferes with the Wi-Fi, the UI seems to freeze every time someone calls you and the system language changes to Russian every 35 days. Suddenly, it hits you. Whilst the components all work on their own, you forgot to check if they work *together*. You forgot to do integration testing.

In the same way that integration testing builds on unit testing to provide some assurance that your tested components work as a cohesive whole, we need some way of formally verifying the entire system. One immediate thought is to stick all the component parts together, then using model checking to look at the holistic system. But this resulting system could be (and often is) far too large to model check. Whilst model checking is a very powerful tool, it is not the best way of reasoning about systems with lots of different components — there are simply too many ways these components can interact. Moreover, given the aims of each component, some executions of the system may be theoretically possible, but may never actually happen in practice.

A key observation is that if we have multiple agents, a given agent does not have to

worry about the other agents taking arbitrary actions — only rational ones. This is where *rational verification* [95, 102] comes into play. In short, rational verification does not ask if certain properties hold unconditionally — it asks if certain properties hold under the assumption that each agent is acting rationally.

Given the meteoric rise of machine learning over the last ten years, and given the increasing use of robotics across many industries, it is clear that intelligent multi-agent systems will become prominent in the years to come. Thus, being able to reason about them effectively will not only be important, but will be crucial in making sure artificial intelligence and its related technologies are safe, efficient, and fair.

1.2 Literature review

Model checking, logic and automata

As stated previously, the model checking problem is of crucial importance, and will be our jumping-off point. In terms of representing computational systems in a finite, expressive way, the two main methods of doing so are with *labelled transition systems* [128] and *Kripke structures* [132]. These two models are isomorphic in the sense that one can easily be encoded as the other. Broadly speaking, these models are graphs that have been labelled in some way, encapsulating the core features of some system. If encoded properly, the execution of the system corresponds to an infinite walk around the graph. The main difference between the two models is that a labelled transition system has sets of atomic propositions attached to each of its transitions, whilst a Kripke structure has these atomic propositions attached to its states instead. Whilst you can work with either, it is more common to work with Kripke structures in the context of model checking.

Pnueli [1977] introduced *linear temporal logic* (LTL) as a way of reasoning about potential behaviours of concurrent programs, and Clarke and Emerson [1981] introduced *computation tree logic* (CTL) for the same purpose. The semantics of LTL are defined with

respect to infinite sequences of subsets of propositional variables, whilst we interpret CTL relative to infinite trees labelled with subsets of propositional variables. Given this, LTL and CTL formulae can also be interpreted in the context of Kripke structures and as such, can be used to describe properties that we may or may not want to hold in the execution of a program.

Whilst each logic has its strengths and weaknesses, it is generally accepted that LTL is easier to write specifications in, whilst CTL has nicer computational properties[2] [204]. There are, of course, many other logics one can work with, but they tend to be extensions/modifications of LTL and CTL [9, 14, 115, 174]. Additionally, as reflected in the tooling available [73, 120], LTL and CTL are the two most widely used, de facto logics in classical model checking. With that having been said, there are two more logics that are at least worth mentioning — CTL* [89], which is an extension of CTL in which every LTL formula can also be defined[3] and the modal μ-calculus [88], which is a very expressive logic, in which CTL*, and thus also LTL and CTL, can be embedded.

With this in place, we can state the model checking problem — given a Kripke structure, and an LTL/CTL formula, does the formula hold within this structure? For LTL, this decision problem is **PSPACE**-complete [146, 193] and for CTL it lies in polynomial time [74]. This does *not* imply, however, that model checking CTL is 'easier' than model checking LTL. It is relatively easy to show that CTL formulae are at least as long as their equivalent LTL formulae, and under the assumption that $\mathbf{P} \neq \mathbf{NP}$, there exist LTL formulae for which the smallest equivalent CTL formulae are exponential in size of the LTL formula [34]. With that having been said, there is a lot of 'room' between **P** and **PSPACE**, and many CTL formulae are not actually that much more verbose than their equivalent LTL formulae. We also note that the corresponding model-checking decision problems for CTL* and the modal mu-calculus are also **PSPACE**-complete, and so are no harder

[2] This comes with a major disclaimer, which is coming up in the next paragraph.

[3] This cannot be done in CTL — that is, there exist LTL formulae which define behaviours for which there is no equivalent CTL formula.

than model-checking LTL.

Finally, as mentioned in Section 1.1, there are mature, open-source, model-checking tools which have seen real-life, industrial usage [63, 71, 185, 186]. Without putting any particular emphasis on any one tool, SPIN [120] and NuSMV [73] are both popular model checkers for temporal logic formulae.

The model checking problem has a privileged position in computer science, and for good reason — it gives you the guarantee that your program adheres to a provided formal specification. However, to be able to use model checking procedures, you need something to model check; that is, a program of some sort. As established in the opening paragraph of this chapter, programming (correctly) is hard. One can go back and forth, writing a program, putting it through a model checker, having it rejected, rewriting and resubmitting it to the model checker and so on, but again, this is far too 'manual'. Instead, would not it be nice if we could somehow automatically write a program which adheres to a given specification?

Almost by magic, this is where the *realisability* (sometimes called 'implementability') and *synthesis* problems come in. Church was one of the first to formalise these problems: informally, given a binary relation over inputs and outputs (with certain constraints), does there exist, and can one construct, a function from inputs to outputs such that each (input, output) pair generated by the function adheres to the binary relation [72, 136]? Asking if such a function exists is called realisability, whilst asking to construct such a function is called synthesis. The former is a decision problem, and the latter a function problem, and our attention will be largely focussed around realisability. However, we will use the term 'synthesis' as a casual term for realisability/implementability throughout, as this conflation of terminology is fairly standard in the literature; moving forward, exactly what we mean when we talk about synthesis should be apparent from the context.

It is a small step from this, to define the binary relation in terms of a logical formula over the inputs and outputs, and ask instead if there exists some program which mod-

els the logical formula on all possible inputs. This is exactly what Pnueli and Rosner [1989] and Rosner [1991] did in the context of LTL, and Clarke and Emerson [1981] considered this problem with CTL formulae. In the setting of CTL, the realisability problem is **EXPTIME**-complete [136], whilst for LTL, it is **2EXPTIME**-complete [178].

One more topic we ought to touch on is that of ω-automata [175]. Whilst they may not exist in the forefront of our discussions, they are the foundation of many model checking algorithms [205]. The natural way to reason about LTL formulae is to convert them to Büchi automata [129, 206] and CTL formulae can be understood by converting them to alternating tree automata [203]. Additionally, model checking algorithms may introduce different types of automata if their computational properties lend themselves to the task at hand [95, 111, 156].

Finally, one formalism we will use throughout this thesis is that of *Emerson-Lei conditions* [15]. Originally used to facilitate 'fair' CTL model checking, it has since seen adoption in other areas, including being used as a general syntax to specify the acceptance conditions of arbitrary ω-automata [33].

Computational game theory

Game theory has a number of flavours, but the one that we are most interested in is that of *computational game theory*. That is, reasoning about the computational properties, such as decidability and complexity, of the algorithms and decision problems related to games.

It is fairly uncontroversial to claim that game theory in its modern form began with von Neumann [1928], who proved the minimax theorem, and together with Morgenstern, later provided the foundational text for the field [165]. This was swiftly followed up with Nash's contributions [161, 162], where he introduced the concept of the Nash equilibrium, and provided certain existence theorems for it. Whilst it was natural to first study two-player, zero-sum games (that is, games where one player's payoff is the negative of the other's), the theory quickly expanded to include multiplayer, general-sum (sometimes

called 'non-zero-sum') games[4].

Whilst game theory was initially seen as a set of tools in economics, its value was quickly appreciated in other fields, such as in biology [78, 151] and in social science [32]. However, for the remainder of this subsection, we will focus on one particular corner of game theory: that of turn-based, two-player, zero-sum, infinite games on finite graphs. The study of these initially dropped out from Büchi's study of the sequential calculus [57] and together with his PhD student, Landweber, they proved some fundamental results about games on graphs [58, 141]; the first modern, comprehensive treatment of this topic was by McNaughton [1993].

Infinite games on finite graphs are immediately an appealing model for many reasons. Traditional parlour games such as chess and checkers can be encoded in this model, and as these games never terminate, they are also useful for modelling always-live systems. Additionally, many multiplayer, general-sum scenarios can either be mapped to these infinite games on graphs, or we can reason about such scenarios via these games [50, 202]. Indeed, whilst in this thesis we will be concerned primarily with concurrent, multiplayer, general-sum games (coming up in the next subsection), we will see that 'under the hood', we will almost always be appealing to results from the turn-based, two-player, zero-sum setting.

Another attractive feature of these games is that for certain winning conditions, the structure of the game is closely related to certain ω-automata, and techniques from one domain can be applied to the other, and vice versa. As far as I am aware, this relationship is usually taken for granted, but Perrin and Pin [2004] offers a very readable exploration and some insight into the ideas involved.

As computer scientists and computational complexity theorists, we are largely interested in one primary question relating to infinite games on finite graphs: given a class of

[4]Indeed, to the best of my understanding, Nash's papers were among the first to consider non-cooperative, multiplayer games in the form we study them today; von Neumann and Morgenstern did consider multiplayer, general-sum games, but under assumptions which reduced them to two-player, zero-sum games. See [1] for a further discussion.

Winning condition	Player 1 wins when…
Reachability	They reach a given state in the game graph
Büchi	They visit a provided set of states infinitely often
co-Büchi	They visit a provided set of states finitely often
Rabin	Given a set of pairs of sets of states, for some pair, they visit the first element finitely often and the second element infinitely often
Street	Given a set of pairs of sets of states, for all pairs, they visit the first element finitely often or the second element infinitely often
Parity	Given a natural number 'colouring' of the states, the greatest colour visited infinitely often is even
Mean-payoff	Given a weight function on the states of the game, the mean-payoff (a sort of infinite average) of the weights encountered is positive
Mean-payoff parity	Given a parity condition and a weight function, the parity condition is satisfied and the mean-payoff of the weights encountered is positive
Muller	Given a set of sets of states, for some set, they visit the states within it infinitely often, and the remaining states finitely often

Table 1.1: Informal explanations of various winning conditions in games.

winning conditions, what is the computational complexity of deciding whether player 1 has a winning strategy in a given game? One may also be interested in algorithms which may actually be used to solve these games in practice, but we will be more interested in the former question. In terms of objectives for the first player, a wide range of representations have been studied. In Table 1.1, we give informal explanations of various winning conditions in two-player, zero-sum games (the necessary formal definitions will come later), before discussing their history and the complexities of their associated decision problems in the coming paragraphs.

From a qualitative perspective, McNaughton [1993] provided relatively general results, showing that the problem of determining the winner in games with Muller objectives is **PSPACE**-complete. Hunter and Dawar [2005] built on this by considering games with various succinct representations of Muller objectives, and showed that determining the winner in these games remains **PSPACE**-complete. For Büchi and co-Büchi games,

the fact that they can be solved in polynomial time seems to fall out from implicitly from the analysis in [152], but it is explicitly explored (and improved upon) by Chatterjee, Henzinger and Piterman [2008]. Immerman [1981] established the **P**-completeness of reachability games as a by-product of his studying of descriptive complexity. Emerson and Jutla [1988] considered games with Rabin and Streett objectives, and showed that determining the winner in these games is **NP**-complete and **co**-**NP**-complete respectively. Finally, Zielonka [1998] was one of the first to formally consider parity games, developing a recursive algorithm for solving them.

In quantitative terms, Ehrenfeucht and Mycielski [1979] introduced mean-payoff objectives and demonstrated that memoryless strategies are sufficient for players to act optimally in these games (that is, to achieve their best possible payoff). This was followed up by Zwick and Paterson [1996], who gave complexity results as well as algorithms for calculating the 'value' (*i.e.* the greatest payoff the first player can force) of a mean-payoff game. In particular, the problem of determining if the value of a game is a given number lies in **NP**∩**co**-**NP**. Jurdzinski [1998] then proved that mean-payoff games can be solved in **UP**∩**co**-**UP**, before demonstrating a straightforward polynomial time reduction from parity games to mean-payoff games, from which it follows that the corresponding decision problem for parity games also lies in **UP** ∩ **co**-**UP**. Building on these ideas, Chatterjee, Henzinger and Jurdzinski [2005] looked at mean-payoff parity games, and Velner et al. [2015] considered multidimensional mean-payoff objectives in two-player games.

Additionally, for a reasonably wide class of winning conditions, there is a very general, powerful result (the Borel Determinacy Theorem) that shows these games with these conditions are determined [150]; that is, for every instance of a game, one player has a winning strategy.

All of the above games are deterministic, but it is worth noting that there has also been a considerable amount of work done on stochastic games with ω-regular objectives [12, 13, 64, 68].

Finally, we call back to our previous subsection where we spoke about synthesis of LTL and CTL formulae. Whilst we framed this problem in the context of model checking, it can also be thought of as an infinite game on a finite graph. In the case of LTL, one can think of a Kripke structure capturing the 'state of the world' as a game graph, where one player can be thought of as a program, trying to achieve some LTL goal, whilst the other can be thought of as an adversarial environment, attempting to outfox the program. The synthesis problem can then simply be asked as 'does there exist some winning strategy for player 1?'. Mathematically, we have done nothing new here, but this viewpoint will be crucial to have in mind as we progress throughout the remainder of this chapter.

Multi-agent systems

In our discussions of both games and model checking so far, we have focussed on the paradigm of 'one player/system against the world'. In two-player, zero-sum games, we have a player trying to achieve a goal, and the other player trying to make sure this does not happen. Within model checking, we try to verify that the system in question satisfies a specification, no matter what transitions the system makes. But this is not an accurate reflection of reality. In a lot of scenarios, there are multiple entities and we would like to understand how they interact. Thus, we look to the domain of multi-agent systems [192, 209].

Multi-agent systems is a rich, diverse field, with a spectrum of different approaches. Subfields include distributed planning, multiplayer games, learning algorithms, communication, voting theory, mechanism design, auctions, coalitional games and logical reasoning [192]. However, we are primarily interested in multi-agent systems from two perspectives: games and logics.

Modelling concurrent and multi-agent systems as games in which players interact by taking actions in pursuit of their preferences is an increasingly common approach in both formal verification and artificial intelligence [10, 20, 117]. One widely adopted semantic

Game theory	Model checking
Game	System
Player	Agent
Outcome	Trace/execution

Table 1.2: Terms from game theory and model checking that we deem to be equivalent.

framework for modelling such systems is that of *concurrent game structures* [20]. Such structures capture the dynamics of a system — the actions that players can perform, and the effects of these actions. On top of this framework, we can introduce additional structure to represent each player's preferences over the possible paths of the system. We will expand on what such preferences might look like momentarily.

Given the central role of game theory in multi-agent systems, we will now start blending terminology from the game-theoretic and model-checking literatures, depending on what interpretation we wish to emphasise in the moment. Table 1.2 summarises terms we generally regard as equivalent.

From a logical perspective, there are two interesting strands for us in multi-agent systems. The first is that of *coalition logic*, introduced by Pauly [2002], of which there is an abundance of extensions [4, 5, 7, 163]. There is also *alternating time temporal logic* (ATL) and its extension ATL*, both introduced by Alur, Henzinger and Kupferman [2002]. On the face of it, these two logics are very similar — they both describe properties pertaining to what coalitions can achieve in an infinite game. However, they have two different underlying models — coalition logic is defined over *game frames* and ATL/ATL* is defined over concurrent game structures. Whilst both logics are widely used, ATL and ATL* are far more common in the model checking community, along with concurrent game structures. However, we should not view coalition logic and ATL as opposing viewpoints: one can show that ATL and Coalition Logic are semantically equivalent, and that Coalition Logic is isomorphic to a fragment of ATL [100].

ATL is very similar to CTL, but instead of quantifying over paths, one quantifies over

sets of agents. Also, ATL* is to ATL as CTL* is to CTL, allowing the binding of LTL formulae over coalitions of agents (*i.e.* we can write statements such as 'the coalition C can force the LTL formula φ to be true'). The model checking problem for ATL is **P**-complete, whilst the same problem for ATL* is **2EXPTIME**-complete [20]; thus, model-checking ATL is no harder than model-checking CTL, but whilst you 'only' gain an exponential penalty in complexity when going from model-checking CTL to CTL*, you gain a doubly-exponential penalty when going from ATL to ATL*.

An important extension of ATL* is Strategy Logic (SL). Introduced by Chatterjee, Henzinger and Piterman [2010] and expanded by Mogavero, Murano and Vardi [2010] and Mogavero et al. [2014], SL goes a step further. Instead of simply quantifying over agents, it quantifies over strategies, which you then bind agents to. Strategy Logic is extremely expressive, allowing you to express quite extravagant properties of a game, but it also comes with a lot of complexity — the model checking problem for SL is decidable but non-elementary. With that having been said, there are large fragments of SL which have more desirable computational properties [38, 155].

Finally, we mention one more important topic in multi-agent systems: that of *reactive modules* [19]. Reactive modules offer its own language for encoding interacting agents. This model is very expressive and has a lot of neat properties (such as compositionality and succinctness). One can then take an ATL formula, and ask if it holds over a set of reactive modules. Closely related to this is the *open systems/module checking* framework of Kupferman and Vardi [1996, 1997]. Open systems are defined in contrast to *closed systems*, which can be thought of the traditional objects of model-checking: systems entirely defined by their state space and their internal transitions. Open systems are instead affected by their environment, and need to be reasoned about differently. Whilst we will not comment on this further, this acts as a natural stepping stone to our next section.

Rational verification

Suppose for a moment, that you yourself lie in a multi-agent system. How should you act to achieve your goals? Should you act cautiously, assuming the worst in others, afraid they may minimax you? Often, people's goals are at least partially aligned with our own. Even if they are not, it is rare for a system to be truly zero-sum. Thus, you should not worry about other agents taking arbitrary actions that might prevent you from achieving your ends — they are too busy trying to achieve their own goals. Thus, instead of trying to understand the system under all possible behaviours of the agents, it seems more natural to ask how the system performs when all the players are behaving *rationally*.

Now whilst the definition of rationality is up for debate, it is clear that this lies firmly in the domain of game theory. Specifically, we will need two sets of components: games, to model our systems, and solution concepts, to define rational behaviour within them. The game models we work with will vary, but will generally map directly from the system in question: they will consist of some players, who can take actions to try and effect change in the system, each of whom will prefer different executions of the system over others. To capture these desires over different outcomes, we equip our players with succinct representations of their preferences, called preference relations; these will look like the winning conditions encountered when we discussed two-player games before, but each tweaked for the multiplayer setting. These games are played by each player choosing a strategy, which when combined, lead to an outcome. The question is then, how should each player choose their strategy? One answer is given by solution concepts.

There are a multitude of solution concepts, but I will focus on two in particular: the Nash equilibrium [161, 162] from non-cooperative game theory, and the core [99] from cooperative game theory. The Nash equilibrium consists of those strategy profiles from which no single player would want to deviate — they can do no better given everyone else's actions. The core takes a more coalitional view: it comprises those strategy profiles from which no coalition can deviate and be better off, regardless of how the remaining

players respond. Each of these solution concepts has its strengths and weaknesses; whilst Nash equilibria are natural, and intuitive, they only account for unilateral deviations and often have counterintuitive properties[5]. On the other hand, the core has a more complicated formulation, but allows for multilateral deviations, with the remaining players having an opportunity to respond, and captures the possibility of coalitions being able to form (exogenous) binding agreements. Both of these solution concepts have many refinements which attempt to fix their deficiencies: for instance, strong Nash equilibria [26, 27] allow for multilateral deviations (whilst still not allowing the remaining players to respond, differentiating it from the core), and stable sets [165] are supersets of the core which strengthen the conditions on what constitutes a beneficial deviation for a coalition. Moreover, one can apply the notion of *subgame perfection* to each of these solution concepts, with a subgame perfect Nash equilibrium/core member being a Nash equilibrium/core member in every subgame of the game, ensuring that players do not make non-credible threats. We will not explore these refinements any further in this thesis, and will instead focus on the 'vanilla' versions of these solution concepts[6]. Moreover, rather than continue into an extended discussion on the merits, pitfalls and interpretations of various solution concepts, we shall move on, and take them for granted as models of rational behaviour. I will informally refer to strategy profiles which adhere to one of these solution concepts as equilibria, but will be careful to specify which solution concept I am talking about when presenting my results.

However, it is well-known in the game theory literature that such equilibria may have undesirable properties; for example, an equilibrium may visit dangerous system states, or lead to a deadlock. Thus, one may also want to check if there exist equilibria which satisfy some additional desirable computational properties associated with the game. This decision problem — that is, determining whether a given property is satisfied on some

[5]Such as in the prisoner's dilemma, where the (Defect, Defect) strategy profile is the only Nash equilibrium of the game, but it is not a socially optimal outcome.

[6]Although, we will very briefly consider a relaxation of the Nash equilibrium at the end of Chapter 4 for computational reasons.

(or every) equilibrium of a given multiplayer game — is known as *rational verification* [95, 102, 211].

Much has been written about rational verification [2, 56, 95, 102, 103, 110, 113, 211], but there is a common set of questions which are usually asked in each guise it takes: the non-emptiness, the existence and the universality problems[7]. These ask, respectively, whether a given game has some equilibrium; whether a given game has an equilibrium which adheres to a given property; and whether all equilibria of a given game adhere to a given property.

There are a number of qualifications deliberately missing from the above description — what the model of the game is, how the agents' goals are defined, what form the property takes and what is means for a solution concept to model such a property. Given the general nature of the above questions, we can study them with a variety of game models, preference relations, solution concepts and specification languages for the property. For instance, Fisman, Kupferman and Lustig [2010] studied Nash equilibria in a game model similar to concurrent game structures where both the player goals and property specification are given by LTL formulae. Gutierrez, Harrenstein and Wooldridge [2017] also looked at Nash equilibria in the model of simple reactive module games with LTL and CTL objectives and specifications. Other work has looked at simpler, more concise game models such as iterated Boolean games [101].

It is also worth noting that whilst the emptiness, existence and universal questions are all important and interesting in their own right, in *most*[8] scenarios, it is sufficient to study just one of them, as they (in general) can be reduced to one another [96]. Culturally, it is most common to consider the existence problem over the other two.

It should be apparent that the problem of rational verification is a natural one to study.

[7] These terms are completely non-standard, but calling them by their usual names, in decision problem notation, is more formal than I want to be in this chapter. I will introduce the proper terminology, along with formal definitions, in the next chapter.

[8] This is a very broad generalisation. It is important to note that in certain situations, deciding the existence problem can be more difficult than deciding the non-emptiness problem, as checking a specification can add a lot of overhead to the problem.

However, in its original setting of Nash equilibria with LTL goals and properties, the decision problem is **2EXPTIME**-complete [101, 102]. This is a complexity class that can only be described as 'wildly intractable'. Therefore, we are motivated to look at preference relations and specifications which induce lower complexities in the rational verification framework, whilst still being expressive and natural. There are a few routes one can take with this.

However, before exploring these different avenues, it is crucial to differentiate between the different formalisms of strategy that one can use in rational verification (as well as in multi-agent systems/iterated games in general). Informally, a player strategy in its most general form is a function that maps what has happened in the game so far to a legal action for that player. But what do we mean when we say 'what has happened in the game so far'? Indeed, this is a subtle issue — do we mean:

- the actions taken by each of the players;
- the states of the game visited; or
- in a setting like LTL, the labels on the states encountered?

Gutierrez et al. [2019] helpfully disambiguates these three cases by calling them *computations*, *runs* and *traces* respectively, and we shall use these terms for the remainder of this section. It should be apparent that strategies based on each these models offer differing amounts of information to the players — computation-strategies offer more information to the agents than run-strategies, which in turn offer more information than trace-strategies. As such, determining if a game has a run-strategy Nash equilibrium requires fundamentally different techniques than determining if it has a computation-strategy Nash equilibrium.

The assumption that strategies have visibility of actions is certainly a common one within game theory (for instance, given their structure, this arises naturally in turn-based games), as well as within rational verification [95, 102, 202, 211]. However, there is a

rich vein of work which uses run-strategies [20, 49, 54, 154, 156] and some would argue that this offers a more accurate model of a multi-agent system. One notable highlight is that of Bouyer et al. [2011], who introduced the *suspect game* construction to transform the (run-strategy) Nash equilibria of multi-player concurrent games into winning strategies of some two-player turn-based game: intuitively, the resulting two-player game has one player proposing an action profile on each turn of the game, progressively trying to develop a Nash equilibrium, whilst the other player tries to prove the proposed strategy is not a Nash equilibrium. Additionally, Gutierrez et al. [2019] explored the bisimulation-invariance (or lack thereof) of Nash equilibria under the different models of strategies, showing that this property holds for some of these models, but not in others. The takeaway is clear — different formalisms of strategies induce radically different behaviours within games, and resultingly, in their solution concepts.

We can now proceed and consider alternatives to the 'vanilla' rational verification setup. Possibly the most straightforward thing to do is consider some fragment of LTL. Various fragments of LTL are very well studied [21, 198] and the decision problems relating to them are much more computationally amenable. Gutierrez, Harrenstein and Wooldridge [2015] considered games where all the players have propositional safety goals — that is, LTL goals of the form $\mathbf{G}\,\varphi$, where φ is some propositional formula. In this case, the existence problem with Nash equilibria is **PSPACE**-complete. Additionally, Gutierrez et al. [2019] studied games with GR(1) [42] goals and specifications. Here, the existence problem is **PSPACE**-complete when you use GR(1) for player goals and LTL for the property specifications, and lies in **FPT** (fixed parameter tractable) [82] when you use both GR(1) for player goals and for specifying the property.

One may ask 'well, what about using CTL instead?'. The story here is no better — with CTL goals and specifications, the existence problem for Nash equilibria is **2EXPTIME**-hard [102]. So what if we forget temporal logic and look at different preference relations altogether? Bouyer et al. [2015, 2016] gave the players ω-regular objectives and looked at

the non-emptiness problem for these games, obtaining a range of complexity results from **P**-completeness up to **EXPTIME** membership.

We can also consider refinements of the Nash equilibrium, such as dominant strategy equilibria [181], strong Nash equilibria [26, 27] or subgame perfect Nash equilibria [169]. In the context of iterated Boolean games, the existence problems for dominant strategy equilibria and subgame perfect Nash equilibria have been shown to be **2EXPTIME**-complete [101].

As you might have inferred, most existing work has focussed on rational verification in the context of Nash equilibria, but this is certainly not the only solution concept that game theory has to offer. Bruyère, Raskin and Tamines [2021, 2022] considered rational verification in the context of a novel solution concept based on the idea of Pareto optimality [24, 25, 79, 170, 171, 172]: one player is trying to achieve their goal, whilst the other player has a set of objectives, of which they are trying to achieve some Pareto-optimal combination of them.

We can also look to cooperative game theory for inspiration. For example, the core [99, 169] is probably the most widely used solution concept in cooperative game theory and we can adapt and use it in the setting of rational verification. To do this, we define a 'beneficial deviation' as a change in strategy for a coalition, that no matter what the remaining players do, each player in the coalition is strictly better off. The core then consists of those strategy profiles from which no coalition has a beneficial deviation. Gutierrez, Kraus and Wooldridge [2019] worked with concurrent game structures with LTL goals and specifications and show that the existence problem for the core is **2EXPTIME**-complete. They show this by establishing that the existence of the core can be characterised with a simple ATL* [20] formula.

Finally, it is worth noting that all the games considered so far have been deterministic with pure strategies and perfect information. Filiot, Gentilini and Raskin [2018] studied rational verification with imperfect information, by considering the Nash existence prob-

lem with parity objectives, and shows that for two players, the relevant decision problem is **EXPTIME**-complete, but undecidable for three or more players. Gutierrez, Perelli and Wooldridge [2018] showed a similar result in the context of reactive module games; they prove that the Nash non-emptiness problem for reactive module games with imperfect information is **2EXPTIME**-complete for two players, and undecidable for three or more players. They also show that you retain decidability if you restrict your model of strategies, and provide complexity results for these settings.

A lot of work has also been done in model checking stochastic games: Gutierrez et al. [2021] showed that the existence problem for both Nash equilibria and the core in concurrent stochastic games with LTL goals and specifications is **2EXPTIME**-complete, whilst Kwiatkowska et al. [2018] considered the problem of determining if rPATL (probabilistic ATL with rewards) specifications hold in turn-based or concurrent games, and Kwiatkowska et al. [2018] looked at the problem of determining if those same specifications hold in the equilibria of a game.

It should be clear there is a lot of flexibility in the setup of the rational verification framework. However, all of what we have considered so far has been centred around games with qualitative goals, where agents are largely unrestricted in their choice of actions. Introducing quantitative features to games can help them model real-world systems more closely, and we will explore existing work on this topic in the following subsection.

Quantitative aspects of games and logics

So far, I have only mentioned purely qualitative settings. One might ask if rational verification can be studied in a quantitative context, and indeed it can. There are two notable formalisms within the realm of resource-bounded two-player games on graphs: mean-payoff games [87, 213] and energy games [48, 62]. As mentioned previously, in mean-payoff games, two players traverse a graph with numerical weights on the states, with

the first player trying to maximise some sort of infinite average of the weights visited over time, whilst in energy games, the first player is aiming to keep their running total of weights visited positive. Despite the superficial differences, these two models are log-space reducible to one another [48].

Mean-payoff games have been generalised to include multiple players [202] in a concurrent game structure, where each player is given a weight function over the states, and they wish to maximise their respective mean-payoffs. In these games, Ummels and Wojtczak [2011] asked if there exists some Nash equilibrium where each player's payoff lies between a certain threshold; this is **NP**-complete. Brice, Raskin and van den Bogaard [2022] went on to show that the same problem remains **NP**-complete for subgame perfect Nash equilibria. If one uses LTL to specify a property in a mean-payoff game, the Nash existence problem is **PSPACE**-complete; if you use GR(1) specifications, then it remains **NP**-complete [110]. Additionally, both two-player mean-payoff and energy games have been extended to include multiple weights on an edge [67, 207].

However, balancing quantitative and qualitative goals and properties is not always as straightforward — for example, in two-player, zero-sum, mean-payoff parity games, where the first player gets their mean-payoff if some parity condition is satisfied, and $-\infty$ if not, optimal strategies may require infinite memory [65]. Thus, given the standard transformation from non-deterministic Büchi automata to deterministic parity automata [176], the outlook is not promising for combining LTL and mean-payoff objectives. Despite this, Gutierrez et al. [2017, 2020] looked at games with lexicographic preferences, where the first component is either a Büchi condition or an LTL formula, and the second component is some mean-payoff objective. Rather than studying the existence problem as it is usually stated, they consider a variant of it — the problem of whether there exists some finite-state, strict ε-Nash equilibrium. These extra qualifiers are brought about precisely because optimal strategies in mean-payoff parity games require infinite memory. In the case where the Büchi condition is the first component of the lexicographic preferences, the

stated decision problem is **NP**-complete, and in the LTL case, the problem is **2EXPTIME**-complete. Thus, despite the relaxation of the solution concept, we do not gain any benefits computationally, but equally, we do not see a blowup into **3EXPTIME** or beyond.

Many logics have been developed for reasoning about resource-bounded systems. One common approach is to extend a logic such as coalition logic, or ATL/ATL*, to include quantitative assertions. Alechina et al. [2011] introduced resource-bounded coalition logic, an extension of coalition logic, within which one can make statements about coalitions being able to achieve a formula under a given resource bound; Alechina et al. [2010] extended ATL similarly. Alechina et al. [2018] defined RB ± ATL(*), two logics which extend ATL/ATL* in the same way as the previous two approaches, but which also allow resource production, as well as consumption. A similarly motivated logic, QATL*, was introduced by Bulling and Goranko [2013]; QATL* is ATL* extended with statements about average and discounted payoffs achieved over infinite paths. QATL* is undecidable, but some of its fragments restore decidability. Finally, Boker et al. [2014] introduced LTL$^{\mathsf{Lim}}$, an extension of LTL where one can make assertions about mean-payoff values over weighted Kripke structures; they then go on to show that model checking in this logic is **PSPACE**-complete — no harder than model checking for LTL.

Other work has been done to introduce non-dichotomous quantitative preferences to rational verification. Kupferman, Perelli and Vardi [2016] introduced *Objective LTL* (OLTL) as a format for specifying player goals and system properties. An OLTL formula consists of a tuple of LTL formulae, along with a function which maps 0–1 tuples of the same length to integers. In a given execution of a system, some of the LTL formulae will be satisfied and others will not be, inducing a 0–1 tuple. This tuple is then mapped to some payoff, which each of the agents is aiming to maximise. They then consider the decision problem of 'given a set of agents along with a system player, does there exist a strategy for the system player and a Nash equilibrium for the remaining players, where the system player's payoff is greater than a given threshold?'. This problem is no harder than

the original rational synthesis problem for LTL and is **2EXPTIME**-complete. Building on this, Almagor, Kupferman and Perelli [2018] considered the rational verification problem with LTL[$\mathscr{F}$] [16] objectives and specifications. In short, LTL[$\mathscr{F}$] generalises LTL by replacing the classical Boolean operators with arbitrary functions which map 0–1 tuples to the interval $[0,1]$. Again, the associated decision problem remains **2EXPTIME**-complete. In a similar vein to the LTL^{Lim} logic mentioned above, Bohy et al. [2013] defined an extension of LTL, LTL_{MP}, which can include statements about mean-payoffs, and shows that performing synthesis in the new logic can be done in **2EXPTIME**, which is no worse than synthesis in the LTL setting.

Whilst LTL and other previously described logics reason about properties of a sequence over discrete timesteps, one can generalise this by turning to a continuous model of time. Alur, Feder and Henzinger [1996] introduced *metric interval temporal logic* (MITL), a formalism which builds on top of LTL and can make statements about Boolean-valued functions over continuous time. Maler and Nickovic [2004] then introduced *signal temporal logic* (STL), which adds further structure to MITL to be able to reason about real-valued functions over continuous time on a bounded interval. In the setting of MITL, satisfiability and model checking are **EXPSPACE**-complete [18], whilst Doyen et al. [2009] show that MITL synthesis is undecidable. Complexity results for STL are more sparse, and efforts seem to be more focused around 'property monitoring' [149], a paradigm that Donzé [2013] describes as 'statistical model checking'; traditional verification techniques are generally deemed to be intractable in this setting [80, 149], and I am unaware of any work that has introduced MITL/STL into rational verification.

Thus we see that OLTL/LTL[$\mathscr{F}$] and MITL/STL offer complementary approaches to extending the type of statements LTL can make — LTL formulae can be seen as maps from infinite sequences of states to the Booleans: OLTL/LTL[$\mathscr{F}$] changes the range of this mapping, whilst MITL/STL changes the domain of it.

Bidding games are another rich class of two player games on graphs, where on each

turn, players bid for the right to choose the next move [28]. There are both finite [142, 143] and infinite [29] duration bidding games, each of which can be studied with a number of different winning conditions (such as reachability, parity, and for infinite-duration games, mean-payoff objectives). Within this there are a variety of subtly different bidding mechanisms [31] (such as whether the winning bid is paid to the other player, or to the bank, etc.) each of which gives the game significantly different properties, as well as varying computational complexities for their corresponding decision problems.

Resource allocation games [183] are games in which multiple players have access to a pool of shared resources, which they each select some subset of. Their payoff is then a function of the number of players who chose each resource. One important subset of these games is *congestion games* (this is the case that Rosenthal [1973] originally studied), where players prefer *not* to share resources; these are closely related to potential games [157]. Avni, Henzinger and Kupferman [2020] considered *dynamic resource allocation games*, which are resource allocation games with play repeated for infinitely many rounds. Despite the similarity in name to the model under consideration in my Chapter 4, resource allocation games do not have an immediate technical link to the work and results presented here.

Tools and implementations

As we have shown, there is a vast wealth of literature on model checking in multi-agent systems, as well as in rational verification. One might ask, 'But this is all purely theoretical so far — has anyone actually applied this in practice?'. The answer is yes: there are a handful of tools that have been developed to solve these problems. We do not comment on their performance here, as most tools are either incomparable in their differing scopes, and what analysis has been done, has been rather ad-hoc.

Lily [124], Unbeast [86], and Acacia+ [43, 93] are (unrelated) tools for LTL realisability and synthesis. Lily claims to be the first tool to be able to perform full LTL synthesis, and

uses a 'Safraless'[9] construction [137]. Unbeast uses an approach known as *bounded synthesis* [94], and Acacia+ refines this algorithm by appealing to antichains. The authors of Acacia+ later extended it to handle the previously mentioned LTL extension, LTL_{MP} [44].

MOCHA [22] was one of the first multi-agent model checking tools. It takes as input a description of a set of reactive modules [19] along with an ATL specification and determines if that formula holds.

MCMAS [147] takes as input ISPL (Interpreted Systems Programming Language) code, a custom language inspired by the *interpreted systems* framework [91]. The user can supply a description of the agents, along with a logical formula, written in a logic which is effectively ATL extended with epistemic [91] and deontic [148] modalities. The authors demonstrate that MCMAS is capable of verifying the correctness of non-trivial protocols (such as the standard solution to the n-dining cryptographers problem [69]). MCMAS was later extended to handle the one-goal fragment of strategy logic [61]; this fragment includes ATL*, and allows rich game-theoretic concepts (for instance; see [103] for a logical characterisation of the core in ATL*) to be encoded within MCMAS.

PRALINE [52] can determine if a given multiplayer mean-payoff game (albeit with lim sup rather than lim inf objectives) has a Nash equilibrium, and moreover, can synthesise such an equilibrium. It does this by using the aforementioned 'suspect game' construction of [49, 50], and as such, assumes a model of strategy based on histories of states, rather than those of actions. The input language uses a C-like syntax.

EVE (Equilibrium Verification Environment) [107] is a tool which determines if a given simple reactive module game [102] has a Nash equilibrium which models some specification. It does this via a novel reduction to parity games [111]. Both the goals of the players and the specification are given by LTL formulae.

[9] Safra [187] presented a method for determinising non-deterministic Büchi automata. Whilst this method is computationally optimal, it is generally perceived to be very complex, and not amenable to implementation. This is a problem for LTL synthesis, as the original algorithm [178] relies on the details of this construction. Kupferman and Vardi [2005] later presented an alternative method for performing LTL synthesis, deemed to be more straightforward, and lovingly named a 'Safraless' procedure.

PRISM-GAMES [140] is a system for verifying properties in concurrent stochastic games. The properties are specified by rPATL [138] formulae and PRISM-GAMES is capable of verifying that such properties hold as well as synthesising strategies that can achieve them. Moreover, it also supports verification and synthesis of subgame perfect social-welfare optimal Nash equilibria [139].

Chapter 2

Preliminaries

2.1 Notational conventions

Throughout, we will deal with various mathematical structures, largely in the form of games, and their underlying computational models. For convenience, rather than explicitly stating the components of a structure each time we need to speak about them, we instead just use the canonical notation for the components as functions from the structure to those components. This is a rather wordy description of a simple idea: if we have a structure which we introduce as $M = (X, S)$, then rather than writing 'let $M = (X, S)$ be such that X has this property', we instead just write 'let M be such that $X(M)$ has this property'. To reiterate, the notation used to describe the components of the structure will double up as functions over those structures which yield the corresponding components.

Finally, as stated in the preface, I have tried to be explicit as possible, but some knowledge of standard mathematical notation and shorthands is assumed. If there is a piece of notation that I have used without defining, and its meaning is not apparent from the context, please refer to the glossary in the back matter of this thesis.

2.2 Sets, sequences, graphs, and trees

Let X be a set. Then X^ω is the set of all infinite sequences in X. Thinking of an element $\pi \in X^\omega$ as a function, $\pi : \mathbb{N} \to X$, we use square brackets to index into π; that is, we write $\pi[t]$ to denote the t^{th} element of π. We also use *slice* notation to denote prefixes, suffixes and fragments of infinite sequences; that is, if $m, n \in \mathbb{N}$, then $\pi[m \mathinner{..} n]$ is shorthand for the sequence $\pi[m]\pi[m+1] \mathinner{..} \pi[n-1]$. We write $\pi[\mathinner{..} n]$ for $\pi[0 \mathinner{..} n]$ and $\pi[m \mathinner{..}]$ for the infinite sequence $\pi[m]\pi[m+1] \ldots$. Rather than using the cumbersome term 'infinite sequence' throughout, we will use the terms 'path' and 'run'. If the context is clear what the underlying set is, instead of writing 'a path/run over/of X', we may just say 'a path' or 'a run'. As discussed in Chapter 1, some authors use terms such as 'computation' or 'trace' for certain, specific infinite sequences. However, for the purposes of this thesis, we exclusively use the terms 'path' and 'run', with the understanding that we will always make clear what set such an infinite sequence is over. Additionally, we also use the term 'path' to refer to finite sequences — again, when we use this term, it should be clear from the context whether we are referring to a finite or infinite sequence[1].

In any run over a set X, some elements of X may occur infinitely often. If X is a finite set, then this will definitely happen. Often, we will be interested in these elements, and given a run π, we call, denoted $\text{permset}(\pi)$.

A graph[2] is a tuple,

$$\mathscr{G} = (V, E),$$

where,

- V is a non-empty set of nodes;

[1] Why use two different words, 'run' and 'path' for the same concept? 'Run' is a term used in automata theory and logic, whilst 'path' is a term from graph theory, which implicitly underlies a large swathe of game theory. As we are inhabitants of both lands, it is only natural that we use both terms, and often, we use the respective terms to stress the interpretation we want to use in that moment of time.

[2] Typically, one would use G as the 'standard' graph identifier. However, we use G for different types of games throughout, and at least twice we need to talk about graphs and games concurrently; therefore, we use $\mathscr{G}$ for graphs to avoid any ambiguity.

- $E \subseteq V \times V$ is a set of edges.

We say that a graph $\mathscr{G}$ is undirected if $(v, w) \in E(\mathscr{G})$ implies that $(w, v) \in E(\mathscr{G})$. Given a graph, $\mathscr{G}$, one can form the undirected version of it $\mathscr{G}' = (V', E')$, by setting $V' = V(\mathscr{G})$ and $E' = E(\mathscr{G}) \cup \{(w, v) \mid (v, w) \in E(\mathscr{G})\}$.

Now, whilst runs are good for talking about the behaviours of deterministic systems, we will need trees to reason effectively about non-determinism. A tree is a tuple

$$\mathscr{T} = (\mathscr{G}, \varepsilon),$$

where,

- $\mathscr{G}$ is a graph;
- $\varepsilon \in V(\mathscr{G})$ is a distinguished root node;
- The induced undirected graph of $\mathscr{G}$ is connected and acyclic;
- There does not exist any $v \in V(\mathscr{G})$ such that $(v, \varepsilon) \in E(\mathscr{G})$.

Given two vertices, $v_1, v_2 \in V(\mathscr{G}(\mathscr{T}))$, if $(v_1, v_2) \in E(\mathscr{G}(\mathscr{T}))$, then we say that v_1 is the *parent* of v_2 and that v_2 is the child of v_1. It should be clear that every vertex except for ε has a unique parent, whilst a vertex may have any number of children (including none at all). An X-labelled tree, $\tau = (\mathscr{T}, l)$, is a tree $\mathscr{T}$ along with a labelling function $l : V(\mathscr{G}(\mathscr{T})) \to X$[3]. For both graphs and trees, we say that they are *finite* if their underlying set of vertices is finite, and infinite otherwise.

A run over a set X, as previously described, can be thought of as a degenerate X-labelled tree, and an infinite X-labelled tree can be thought of as a collection of runs over

[3]Similarly to the previous footnote, we would like to use λ as the symbol for our labelling function, but it is already reserved for the labelling function of LTL games. To avoid the implication that a tree labelled with λ is a tree labelled with sets of atomic propositions, we avoid λ in this setting and use l instead.

X. Specifically, we say that a *branch* of a tree $\mathscr{T}$ is a run $v_0v_1 \ldots$ such that $v_0 = \varepsilon$ and for every $i \in \mathbb{N}$, we have that v_{i+1} is a child of v_i.

Finally, as this is a thesis about *quantitative* rational verification, we ought to provide our first taste of quantitativeness[4]: given an infinite sequence of numbers, we need a way of sensibly assigning an 'average' value to it. Let $\beta \in \mathbb{R}^\omega$. The *mean-payoff* of β, denoted $\mathsf{mp}(\beta)$ is defined to be,

$$\liminf_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \beta[i].$$

Closely related is the *mean-cost*, denoted $\mathsf{mc}(\beta)$, defined as

$$\limsup_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \beta[i].$$

It is easy to verify that $\mathsf{mp}(\beta) = -\mathsf{mc}(-\beta)$.

In some literature [207], lim sup is used in place of (but also in addition to) lim inf within the definition of the mean-payoff, giving rise to two separate values: the lim inf version denoted $\underline{\mathsf{mp}}(\beta)$, and the lim sup version denoted $\overline{\mathsf{mp}}(\beta)$. This distinction is particularly important when working in multiple dimensions, but for our purposes, we shall focus solely on the lim inf version, in alignment with Ehrenfeucht and Mycielski [1979], Ummels and Wojtczak [2011] and Zwick and Paterson [1996].

2.3 Game-theoretic notions

Games exist in a wide variety of forms [169], but fundamentally, despite the differences in notation and semantics, they share commonalities. So we can talk about standard game-theoretic ideas in a general sense, then specialise our definitions to the model at hand

[4]The dictionary reassures me that this is indeed a correctly formed word.

when appropriate. A strategic game G is a tuple,

$$G = \left(\text{Ag}, \Omega, \{\Sigma_i\}_{i\in\text{Ag}}, g, \{\succ_i\}_{i\in\text{Ag}}\right)$$

where,

- Ag is a finite, non-empty set of *players* (or agents);
- Ω is a non-empty set of *outcomes*;
- For each player $i \in \text{Ag}$, we have that Σ_i is a non-empty set of *strategies* for player i;
- $g : \Sigma_1 \times \cdots \times \Sigma_{|\text{Ag}|} \to \Omega$ is an *outcome* function, taking a strategy for each player, and producing an outcome;
- For each player $i \in \text{Ag}$, we have that $\succ_i \subseteq \Omega \times \Omega$ is a preference relation. We write it infix-style; that is, when $(\omega_1, \omega_2) \in \succ_i$, we write $\omega_1 \succ_i \omega_2$.

Unsurprisingly, strategies will play a key role in our strategic games, so it will be beneficial for us to introduce some additional notation and terms. A strategy profile is a vector of strategies, $\vec{\sigma} = (\sigma_1, \ldots, \sigma_{|\text{Ag}|})$, one for each player, and the set of strategy profiles is denoted by Σ. If we have a strategy profile $\vec{\sigma} = (\sigma_1, \ldots, \sigma_{|\text{Ag}|})$, we use the notation $\vec{\sigma}_{-i}$ to denote the vector $(\sigma_1, \ldots, \sigma_{i-1}, \sigma_{i+1}, \ldots, \sigma_{|\text{Ag}|})$ and $(\vec{\sigma}_{-i}, \sigma_i')$ to denote $(\sigma_1, \ldots, \sigma_{i-1}, \sigma_i', \sigma_{i+1}, \ldots, \sigma_{|\text{Ag}|})$. Finally, we write Σ_{-i} as shorthand for $\Sigma_1 \times \cdots \times \Sigma_{i-1} \times \Sigma_{i+1} \times \cdots \times \Sigma_{|\text{Ag}|}$. All of these notations can also be generalised in the obvious way to *coalitions* of agents, $C \subseteq \text{Ag}$.

The game should be interpreted as follows: each player i chooses a strategy $\sigma_i \in \Sigma_i$, yielding a strategy profile $\vec{\sigma}$. The outcome of the game is then given by $\omega = g(\vec{\sigma})$. If the players choose different vectors of strategies, then in general, the game will have different outcomes, and players will have preferences over those outcomes, given by the $\succ_i$ relation. This begs the question (which is really the central question of game theory): how should the players act? That is, what strategy should they each choose? Intuitively, one may say

'they should act according to their preferences', but what does this really mean? If a player was alone, they could just choose the strategy which gives them their most preferred outcome[5]. But in games with multiple players, the outcome will (in general) depend on all the players' strategies, meaning that a player may not be able to force a particular outcome to happen. To provide a framework for what rational behaviour looks like in a strategic game, we appeal to the notion of *solution concepts*.

To analyse our games, we make use of solution concepts from both the *non-cooperative* and *cooperative* game theory literatures. With respect to non-cooperative solution concepts, a strategy profile $\vec{\sigma}$ is said to be a *Nash equilibrium* [161, 162] if for all players i and strategies σ_i', we have $g(\vec{\sigma}) \succeq_i g(\vec{\sigma}_{-i}, \sigma_i')$. Informally, a Nash equilibrium is a strategy profile from which no player has any incentive to unilaterally deviate.

In addition to Nash equilibrium, we also consider the cooperative solution concept known as the *core* [99, 103]. While Nash equilibria are profiles that are resistant to unilateral deviations, the core consists of profiles that are resistant to those deviations by coalitions of agents, where every member of the coalition is better off, regardless of what the rest of the agents do. Formally, we say that a strategy profile, $\vec{\sigma}$, is in the core if for all coalitions $C \subseteq \text{Ag}$, and strategy vectors $\vec{\sigma}'_C$, there is some complementary strategy vector $\vec{\sigma}'_{\text{Ag}\setminus C}$ such that $g(\vec{\sigma}) \succeq_i g(\vec{\sigma}'_C, \vec{\sigma}'_{\text{Ag}\setminus C})$, for some $i \in C$. Alternatively, we can define the core in the language of *beneficial deviations*: given a strategy profile, $\vec{\sigma}$, and a coalition $C \subseteq \text{Ag}$, a beneficial deviation is a strategy vector $\vec{\sigma}'_C$ such that for all complementary strategy vectors, $\vec{\sigma}'_{\text{Ag}\setminus C}$, we have that $(\vec{\sigma}'_C, \vec{\sigma}'_{\text{Ag}\setminus C}) \succ_i \vec{\sigma}$ for all $i \in C$. The core is then those strategies profiles which admit no beneficial deviation by any coalition. In traditional cooperative game theory, the core possesses multiple desirable properties: in particular, it is efficient (*i.e.* there is no unutilised value) and no coalition can benefit by deviating. For the core's properties in the context of concurrent games, we build on the work first outlined by

[5] It could be the case that there is no 'most preferred outcome' for a player, if there were, say, an infinitely increasing chain of increasingly preferable outcomes. However, all the outcome spaces we consider in our work are closed and bounded, and we will not encounter such a scenario.

Gutierrez, Kraus and Wooldridge [2019] in the setting of LTL objectives, extending their results to a number of different game models. In particular, we will show that the particular properties of the core are highly contingent on the exact formulation of a game model[6], and will discuss the associated results throughout.

These two approaches differ in that in the non-cooperative setting, agents cannot form binding agreements with other agents: they must make choices and act in isolation. In contrast, in the cooperative setting, it is assumed that agents can form binding agreements to work in teams called *coalitions*; the mechanism for forming such an agreement is outside the scope of this work. We also note that whilst our definition of the core is aligned with its traditional definition, we use a version couched in the language of non-cooperative games, recently introduced by Gutierrez et al. [2019].

Given a game G, let NE(G) denote the set of Nash equilibrium strategy profiles of G, and let CORE(G) denote the set of strategy profiles in the core of G. It is worth noting that if a strategy profile is not a Nash equilibrium, then at least one player can deviate and be better off, under the assumption that the remainder of the players do not change their actions. However, if a strategy profile is not in the core, then some coalition can deviate and become better off, *regardless of what the other players do*. Thus, the core should not be confused with the solution concept of *strong Nash equilibrium*: a strategy profile that is stable under multilateral deviations, assuming the remainder of the players 'stay put' [26, 27]. We will not use strong Nash equilibria in this work, and only mention the concept in order to emphasise how the strong Nash equilibrium is different from the core.

[6] As a 'sneak preview': Gutierrez, Kraus and Wooldridge [2019] proves that the core is always non-empty in LTL games, whilst we show that the core can be empty in mean-payoff games.

2.4 Logics and specification languages

Linear temporal logic

To describe the goals of our agents, we use linear temporal logic (LTL) [177]. Here we give an overview of LTL; for more detail, see, for example, Baier and Katoen [2008].

Let AP be a finite, non-empty set of atomic propositions. The set of LTL formulae over AP is generated by φ in the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \varphi \,\mathbf{U}\, \varphi,$$

where $p \in \text{AP}$. For a set of atomic propositions, AP, we denote the set of LTL formulae over AP by $\mathscr{L}(\text{AP})$. If AP is apparent from the context, we will drop it and write $\mathscr{L}$ only.

The semantics of LTL are defined relative to infinite *runs* of sets of atomic propositions. That is, given a sequence, $\pi \in \mathscr{P}(\text{AP})^{\omega}$, and an LTL formula φ, we say that π *models* φ and write $\pi \vDash \varphi$ according to the following inductive definition:

- If $p \in \text{AP}$, then $\pi \vDash p$ if and only if $p \in \pi[0]$;
- If $\varphi \in \mathscr{L}(AP)$, then $\pi \vDash \neg\varphi$ if and only if it is not the case that $\pi \vDash \varphi$;
- If $\varphi, \psi \in \mathscr{L}(\text{AP})$, then $\pi \vDash \varphi \vee \psi$ if and only if $\pi \vDash \varphi$ or $\pi \vDash \psi$;
- If $\varphi \in \mathscr{L}(\text{AP})$, then $\pi \vDash \mathbf{X}\varphi$ if and only if $\pi[1..] \vDash \varphi$;
- If $\varphi, \psi \in \mathscr{L}(\text{AP})$, then $\pi \vDash \varphi \mathbf{U} \psi$ if and only if there is some $k \in \mathbb{N}$ such that $\pi[k..] \vDash \psi$ and for all $0 \leq t < k$, we have $\pi[t..] \vDash \varphi$.

In addition to the usual propositional shorthands (such as $\cdot \wedge \cdot$ and $\cdot \rightarrow \cdot$), we use $\mathbf{F}\varphi$ to mean $\top \,\mathbf{U}\, \varphi$, and $\mathbf{G}\varphi$ for $\neg\, \mathbf{F}\, \neg\varphi$. One other (non-standard) notation we will use, which

will be useful in an upcoming example, is,

$$\mathbf{Within}_k\, \varphi = \varphi \vee \mathbf{X}\varphi \vee \ldots \vee \overbrace{\mathbf{X} \ldots \mathbf{X}}^{k}\varphi,$$

where $k \in \mathbb{N}$. This **Within** operator can intuitively be thought of as a bounded version of the **F** operator.

Emerson-Lei conditions

Whilst LTL is a standard, well-accepted formalism for specifying path-based properties, it suffers from being rather computationally intractable: as highlighted in Chapter 1, model-checking and satisfiability lie in **PSPACE**, whilst synthesis lies in **2EXPTIME**. One alternative is to use a well-regarded fragment of LTL, such as GR(1) [42]. Instead, we will appeal to Emerson-Lei conditions [15]; their relative simplicity, yet high expressivity, combined with their more reasonable computational properties, make them an attractive alternative to LTL. We will use them both for specifying behaviours of a system (within Chapter 3 and Chapter 5) as well as for specifying individual player preferences (in Chapter 5).

Babiak et al. [2015] used Boolean combinations of atoms of the form Inf(F), where F is some subset of states of a system, to describe acceptance conditions of arbitrary ω-automata. These are, in fact, Emerson-Lei conditions, albeit with a different syntax. We use them here in the context of games, again, with a slightly different (and more general) syntax. Given a set of states, St, the set of Emerson-Lei conditions over St is given by the grammar,

$$\alpha ::= \mathrm{Inf}(s) \mid \neg\alpha \mid \alpha \vee \alpha,$$

where $s \in \mathrm{St}$. Note that our Inf atom is parameterised by states, rather than the sets of states used by Babiak et al. [2015] — for notational convenience, where $F \subseteq \mathrm{St}$ is a set, we

Type	Associated Sets	Emerson-Lei condition
Büchi	$F \subseteq \mathrm{St}$	$\mathrm{Inf}(F)$
Co-Büchi	$G \subseteq \mathrm{St}$	$\mathrm{Fin}(G)$
Gen. Büchi	$(F_k)_{k\in K} \subseteq 2^{\mathrm{St}}$	$\bigwedge_{k\in K} \mathrm{Inf}(F_k)$
Rabin	$(L_i, U_i)_{i\in I} \subseteq 2^{\mathrm{St}} \times 2^{\mathrm{St}}$	$\bigvee_{i\in I} \mathrm{Fin}(L_i) \wedge \mathrm{Inf}(U_i)$
Streett	$(L_j, U_j)_{j\in J} \subseteq 2^{\mathrm{St}} \times 2^{\mathrm{St}}$	$\bigwedge_{j\in J} \mathrm{Fin}(L_j) \vee \mathrm{Inf}(U_j)$
Muller	$(F_k)_{k\in K} \subseteq 2^{\mathrm{St}}$	$\bigvee_{k\in K} \mathrm{Inf}(F_k) \wedge \mathrm{Fin}(\overline{F_k})$

Table 2.1: Traditional ω-regular winning conditions and how they can be expressed with Emerson-Lei conditions.

write $\mathrm{Inf}(F)$ as shorthand for the expression,

$$\bigvee_{s\in F} \mathrm{Inf}(s).$$

We also write $\mathrm{Fin}(F)$ as shorthand for $\neg\,\mathrm{Inf}(F)$, $\mathrm{Inf}(\bar{F})$ for $\mathrm{Inf}(\mathrm{St} \setminus F)$, and we define disjunction, $\cdot \vee \cdot$, implication $\cdot \to \cdot$, and bi-implication $\cdot \leftrightarrow \cdot$ in the usual way. The size of a specification is defined to be the number of atoms it contains.

The semantics of Emerson-Lei conditions are defined relative to infinite runs of states, $\rho : \mathbb{N} \to \mathrm{St}$ in the obvious way:

- If $s \in \mathrm{St}$, then $\rho \vDash \mathrm{Inf}(s)$ if and only if $s \in \mathrm{permset}\big(\rho\big)$;
- $\rho \vDash \neg\alpha$ if and only if it is not the case that $\rho \vDash \alpha$;
- $\rho \vDash \alpha \vee \beta$ if and only if $\rho \vDash \alpha$ or $\rho \vDash \beta$.

Using this notation, we can readily define conventional ω-regular winning conditions, as shown in Table 2.1.

As another example, consider parity conditions. Suppose each state is labelled by a function, $\mu : \mathrm{St} \to \mathbb{N}$, with $\mu(s) \leq m$ for some $m \in \mathbb{N}$, for all $s \in \mathrm{St}$. Given a path, π, we say that the parity condition is satisfied when $\max\{\mu(s) \mid s \in \mathrm{Inf}(\pi)\}$ is even (here, the max can be replaced for min and the even for odd, but this would only result in minor changes to

the analysis). The sets of interest are defined by

$$F_k = \{s \in \mathrm{St} \mid \mu(s) = k\}$$

Then, assuming m is even (the formula for m odd is similar), the parity winning condition can be expressed by the following Emerson-Lei condition:

$$\alpha = \mathrm{Inf}(F_m) \vee (\mathrm{Fin}(F_{m-1}) \wedge (\mathrm{Inf}(F_{m-2}) \vee (\mathrm{Fin}(F_{m-3}) \wedge \cdots \vee (\mathrm{Fin}(F_1) \wedge \mathrm{Inf}(F_0)))))$$

Given that we have proposed Emerson-Lei conditions as an alternative to LTL specifications, it is natural to ask how they compare. The connection between them is given by the following statement:

Proposition 1. *Let G be a game and let α be some Emerson-Lei condition. Then there exists a set of atomic propositions, Φ, a labelling function $\lambda : St \to \mathscr{P}(\Phi)$, and an LTL formula φ such that, for all paths π, we have $\pi \vDash \alpha$ if and only if $\lambda(\pi) \vDash \varphi$.*

Proof. Without loss of generality, we may assume that α is written in conjunctive normal form, that is:

$$\alpha = \bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m} C_{i,j} \right),$$

where each $C_{i,j}$ is an atom of the form $\mathrm{Inf}(F)$ or $\mathrm{Fin}(F)$ for some subset $F \subseteq \mathrm{St}$. We start by introducing a propositional variable p_F for every subset $F \subseteq \mathrm{St}$. Then, for a given state $s \in \mathrm{St}$, we define,

$$\lambda(s) = \{p_F \in \mathrm{AP} \mid s \in F\}.$$

Then, we simply define,

$$\varphi = \bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m} D_{i,j} \right),$$

where $D_{i,j} = \mathbf{G}\,\mathbf{F}\,p_F$ if $C_{i,j} = \text{Inf}(F)$ and $D_{i,j} = \mathbf{F}\,\mathbf{G}\,\neg p_F$ if $C_{i,j} = \text{Fin}(F)$. We claim that for all paths π, we have $\pi \vDash \alpha$ if and only if $\lambda(\pi) \vDash \varphi$.

First suppose that $\pi \vDash \alpha$. Thus, by definition, we have for all $1 \leq i \leq n$ that $\pi \vDash \bigvee_{j=1}^{m} C_{i,j}$. This in turn implies that there exists some j such that $\pi \vDash C_{i,j}$. If $C_{i,j} = \text{Inf}(F)$, then this implies that $\text{permset}(\pi) \cap F \neq \varnothing$. Take any $s \in \text{permset}(\pi) \cap F$. By definition, we have $p_F \in \lambda(s)$ and so we also have $\lambda(\pi) \vDash \mathbf{G}\,\mathbf{F}\,p_F$. But by construction, this implies $\lambda(\pi) \vDash D_{i,j}$.

Similarly, if $C_{i,j} = \text{Fin}(F)$, then we have $\text{permset}(\pi) \cap F = \varnothing$. Thus, for all $s \in \text{permset}(\pi)$, we have $p_F \notin \lambda(s)$ and so we have $\lambda(\pi) \vDash \mathbf{F}\,\mathbf{G}\,\neg p_F$. By construction, this implies that $\lambda(\pi) \vDash D_{i,j}$. Thus, for all i, there exists some j such that $\lambda(\pi) \vDash D_{i,j}$. This implies that $\lambda(\pi) \vDash \varphi$.

Conversely, suppose that $\lambda(\pi) \vDash \varphi$. So for all i, there exists a j such that $\lambda(\pi) \vDash D_{i,j}$. If $D_{i,j} = \mathbf{G}\,\mathbf{F}\,p_F$, then π visits some state $s \in F$ infinitely often. Thus, $\text{permset}(\pi) \cap F \neq \varnothing$, so $\pi \vDash \text{Inf}(F)$. Similarly, if $D_{i,j} = \mathbf{F}\,\mathbf{G}\,\neg p_F$, then $\pi \vDash \text{Fin}(F)$. Either way, we have $\pi \vDash C_{i,j}$. So for all i, there exists some j such that $\pi \vDash C_{i,j}$, implying that $\pi \vDash \alpha$. □

Thus, Emerson-Lei conditions can be seen as being 'isomorphic' to a strict subset of LTL; it is straightforward to come up with LTL formulae that cannot be written as Emerson-Lei conditions — take for instance $\mathbf{G}\,\varphi$, where φ is some non-trivial propositional formula. This quickly highlights how restrictive Emerson-Lei conditions are relative to LTL, but we look to turn this limitation into a strength — as alluded to earlier and as highlighted in Table 2.2, the Emerson-Lei condition formulations of certain key decision problems compared to their LTL counterparts are *significantly* easier in terms of their computational complexity. In conjunction with their expressiveness and their natural interpretation, we see this as a strong motivator to introduce Emerson-Lei specifications into rational verification. As such, we will study a number of decision problems within the rational verification framework, where Emerson-Lei conditions replace LTL specifications in the obvious way, in the hope that this will also yield a reduction in complexity in this setting.

Decision Problem	LTL	Emerson-Lei Conditions
Model-checking	**PSPACE**-complete [146, 193]	in **P** [60]
Satisfiability	**PSPACE**-complete [193]	**NP**-complete [76]
Synthesis	**2EXPTIME**-complete [178, 184]	**PSPACE**-complete [152]

Table 2.2: The relative computational complexities of the important decision problems for LTL and EL conditions.

2.5 Concurrent game structures

A concurrent game structure (CGS) [20] is a tuple,

$$M = (\mathrm{Ag}, \mathrm{St}, s^0, \{\mathrm{Ac}_i\}_{i \in \mathrm{Ag}}, \mathrm{tr}),$$

where,

- Ag and St are finite, non-empty sets of agents and states, respectively, with $s^0 \in \mathrm{St}$ a designated start state;
- For each $i \in \mathrm{Ag}$, Ac_i is a set of actions available to agent i;
- $\mathrm{tr} : \mathrm{St} \times \mathrm{Ac}_1 \times \cdots \times \mathrm{Ac}_{|\mathrm{Ag}|} \to \mathrm{St}$ is a transition function.

We define the *size* of M to be $|\mathrm{St}| \cdot |\mathrm{Ac}|^{|\mathrm{Ag}|}$ — whilst we initially assume we have an explicit, non-concise representation of our concurrent game structures, we will later explore more compact computational structures.

Concurrent games are played as follows. The game begins in state s^0, and each player $i \in \mathrm{Ag}$ simultaneously picks an action $\mathrm{ac}_i^0 \in \mathrm{Ac}_i$[7]. The game then transitions to a new state, $s^1 = \mathrm{tr}(s^0, \mathrm{ac}_1^0, \ldots, \mathrm{ac}_{|\mathrm{Ag}|}^0)$, and this process repeats. Thus, the n^{th} state visited is $s^n = \mathrm{tr}(s^{n-1}, \mathrm{ac}_1^{n-1}, \ldots, \mathrm{ac}_{|\mathrm{Ag}|}^{n-1})$. Since the transition function is deterministic and we have a fixed sequence of actions, a play of a game will be an infinite sequence of states, $\pi : \mathbb{N} \to \mathrm{St}$. On

[7] As discussed in Chapter 1, whilst probabilistic models and randomised strategies are possible in concurrent games, in this thesis we solely considered deterministic, non-randomised behaviour on the part of our agents.

top of concurrent game structures, we can add preferences over runs of states, to transform them into proper games. If we label the states with set of atomic propositions, and then give each player an LTL goal over those atomic propositions, we have LTL games. If instead we add weights to each state, one for each player, and say that players prefer runs with maximise the lim inf of the average of the weights encountered, then we have mean-payoff games. Whilst there is endless variation in terms of adding preferences to concurrent game structures, these two aforementioned games, LTL games and mean-payoff games, will be the most relevant to us (we will formally define LTL games momentarily, and mean-payoff games in Chapter 3).

Strategies

In order to describe how each player acts in a concurrent game, we need to instantiate the concept of a strategy in CGSs. As discussed in Chapter 1, there are several routes we can take with respect to our model of strategies, and whether they are based on histories of actions, states, or labels. Throughout this thesis, we shall work with models that are based on histories of actions — in particular, we assume full visibility of player behaviour to all others players.

Thus, mathematically, a strategy for a given player i can be understood as a function, $\sigma_i : \mathrm{Ac}^+ \to \mathrm{Ac}_i$, which maps sequences, or histories, of actions into a chosen action for player i. A strategy profile $\vec{\sigma} \in \Sigma$ together with a start state s will induce a unique run of states, which we denote by $\pi(\vec{\sigma}, s) : \mathbb{N} \to \mathrm{St}$, as well as a run of actions $\vec{\mathrm{ac}} : \mathbb{N} \to \mathrm{Ac}$, with $\mathrm{Ac} = \mathrm{Ac}_1 \times \cdots \times \mathrm{Ac}_{|\mathrm{Ag}|}$; these sequences are obtained in the following way: starting from s, each player plays $\mathrm{ac}_i^0 = \sigma_i(\epsilon)$. This transforms the game into a new state, given by $s^1 = \mathrm{tr}(s, \mathrm{ac}_1^0, \ldots, \mathrm{ac}_{|\mathrm{Ag}|}^0)$. Each player then plays $\mathrm{ac}_i^1 = \sigma_i(\mathrm{ac}^0)$, and this process repeats infinitely often. Typically, we are interested in runs that begin in the game's start state, s^0, and we write $\pi(\vec{\sigma})$ as shorthand for the infinite path $\pi(\vec{\sigma}, s^0)$.

When considering computational questions surrounding concurrent games, it is use-

ful to work with a *finite* representation of strategies. We use two such representations: *finite-memory strategies* and *memoryless strategies*. A finite-memory strategy for a player, i, is a tuple,

$$\sigma_i = (Q_i, q_i^0, \delta_i, \tau_i),$$

where,

- Q_i is a finite, non-empty set of states with $q_i^0 \in Q_i$ an initial state;
- $\delta_i : Q_i \times \mathrm{Ac}_1 \times \cdots \times \mathrm{Ac}_{|\mathrm{Ag}|} \to Q_i$ is a transition function;
- $\tau_i : Q_i \to \mathrm{Ac}_i$ is an action function.

Thus, finite-memory strategies are finite state machines with output, also known as *Moore machines* [158]. This strategy operates by starting in its initial state, q_i^0, and producing an action $\mathrm{ac}_i^0 = \tau_i(q_i^0)$. Combined with the other players' actions, this yields an action profile $\vec{\mathrm{ac}}^0$. As established above, the game moves a new state based on its transition function tr, whilst player i's strategy moves to the state $q_i^1 = \delta(q_i^0, \vec{\mathrm{ac}}^0)$. This process repeats in a similar way forever. It is not hard to see that if we have multiple finite-memory strategies playing against each other, then the play they generate will be *periodic*: the play must eventually revisit some configuration, and at this point, the game will start to repeat itself. More precisely, any play generated by a collection of finite-memory strategies will consist of a finite non-repeating sequence of states, followed by a finite sequence that repeats infinitely often. Because such a play will be periodic, we can write the path induced on the concurrent game structure as $\pi = \pi[..k]\pi[k..m]^\omega$, for some $k, m \in \mathbb{N}$ with $0 \leq k < m$.

Finally, a memoryless strategy is a strategy that depends only on the state the player is currently in. Thus, memoryless strategies can be expressed as functions $\sigma_i : \mathrm{St} \to \mathrm{Ac}_i$, which simply map states to actions; such functions can be directly implemented as lookup tables, in space $O(|\mathrm{St}|)$. Note that memoryless strategies can be encoded as finite-memory strategies, and that finite-memory strategies are indeed a special case of arbitrary

strategies. Whilst we will work with finite-memory and memoryless strategies, we will use arbitrary strategies by default, unless otherwise stated.

2.6 Rational verification via LTL games

So far, we have spoken around rational verification, giving an informal account in Chapter 1, but it is now time to formally explain the notion. We will explain rational verification in the context of LTL games, as this is the traditional setting in which it is framed in [95, 102]. An LTL game G is a tuple,

$$G = \left(M, \text{AP}, \lambda, \{\gamma_i\}_{i \in \text{Ag}(M)}\right),$$

where,

- M is a concurrent game structure;
- AP is a finite, non-empty set of atomic propositions;
- $\lambda : \text{St}(M) \to \mathscr{P}(AP)$ is a labelling function;
- For each player $i \in \text{Ag}(M)$, we have that γ_i is an LTL formula over AP, called the *goal* of i.

Given a run of states $\pi = s^0 s^1 s^2 \ldots$ over $\text{St}(M)$, we can label it with λ, yielding a run of sets of atomic propositions, $\lambda(\pi) = \lambda(s^0)\lambda(s^1)\lambda(s^2)\ldots$. Then given an LTL formula φ, we can interpret it relative to $\lambda(\pi)$. In particular, we can do this for the players' goals: if for a player i we have $\lambda(\pi) \vDash \gamma_i$, then we say that i is a *winner* under π, and a *loser* otherwise. We then use this to define each player's preference relations: a player i prefers runs in which they are a winner over those in which they are a loser, but is indifferent between runs in which they are winners, and indifferent between runs in which they are losers.

Now, naturally, players should act to try and achieve their goals. But in a given game, they may not be able to achieve their goal by themselves, or there may be multiple, different ways of achieving their goal. Moreover, their actions will affect the satisfaction of the remaining players' goals and vice-versa. Thus, the players need to take the other players' goals into consideration as they reason rationally and strategically.

The natural route to take is to ask about the solution concepts of these games; for us, that is the Nash equilibrium, and the core. Given an LTL game, we want to understand whether any of its strategies are equilibria, and if so, to determine what properties they have. The most natural way of asking these questions is with the notion of decision problems from computational complexity theory. The set of decision problems we consider in the non-cooperative setting is as follows:

Nash-Membership:
Given: An LTL game G and a strategy profile $\vec{\sigma}$.
Question: Is $\vec{\sigma}$ a Nash equilibrium $\vec{\sigma}$ in G?

Non-Emptiness:
Given: An LTL game G.
Question: Does there exist some Nash equilibrium $\vec{\sigma}$ in G?

E-Nash:
Given: An LTL game G and an LTL formula φ.
Question: Does there exist some Nash equilibrium $\vec{\sigma}$ of G which satisfies φ?

A-Nash:
Given: An LTL game G and an LTL formula φ.
Question: Is it the case that φ holds on all Nash equilibria $\vec{\sigma}$ of G?

As discussed in Chapter 1, it is often sufficient just to study one of Non-Emptiness, E-Nash and A-Nash, and we will focus our attention primarily on the E-Nash problem

throughout. Bearing this in mind, in the cooperative setting, the problems we will look at are:

Core-Membership:

Given: An LTL game G and a finite-memory strategy profile $\vec{\sigma}$.

Question: Is $\vec{\sigma}$ a member of the core?

Coalition-SAT:

Given: A concurrent game structure M, a coalition C and an LTL formula φ.

Question: Does there exist a strategy vector $\vec{\sigma}_C$ such that $\rho(\vec{\sigma}_C, \vec{\sigma}_{\mathrm{Ag}\setminus C}) \vDash \varphi$, for all responses $\vec{\sigma}_{\mathrm{Ag}\setminus C}$?

Beneficial-Deviation (Ben-Dev):

Given: An LTL game G, a strategy profile $\vec{\sigma}$, and a strategy vector $\vec{\sigma}'_C$.

Question: Is $\vec{\sigma}'_C$ a beneficial deviation from $\vec{\sigma}$?

E-Core:

Given: An LTL game G and an LTL formula φ.

Question: Does there exist some member of the core $\vec{\sigma}$ such that $\rho(\vec{\sigma}) \vDash \varphi$ holds?

In terms of complexities, we have that Nash-Membership and Ben-Dev are **PSPACE**-complete [101, 103]. The inherent '**PSPACE**'-ness of these questions comes from the **PSPACE**-completeness of the LTL model checking problem [146, 193]. This problem takes a program and an LTL specification and asks whether the program models the specification. As these two decision problems have a strategy profile as an input, they can be seen as analogous to LTL model checking with its program input. However, if instead we remove the strategy profile, our problems start to look more like the problem of LTL synthesis discussed in Chapter 3; this problem takes an LTL specification and asks if there exists some program which models it — this problem is **2EXPTIME**-complete [178, 184]. As such, when we remove the strategy profile from the input, and ask if there *exists* some

strategy profile which is a member of a solution concept which models a specification, we might expect the problems we consider to be **2EXPTIME**-complete. Indeed, this is exactly what we see: the Non-Emptiness, E-Nash and A-Nash problems in the non-cooperative setting, and the Core-Membership, Coalition-SAT and E-Core problems in cooperative setting are all **2EXPTIME**-complete [95, 101, 102, 103].

It is worth noting here one technical detail about representations of strategies. In the E-Nash problem, the quantifier asks if there exists a Nash equilibrium which models the specification. This quantification ranges over all possible Nash equilibria and the strategies may be arbitrary strategies. However, in the Nash-Membership problem, the strategy $\vec{\sigma}$ is part of the input, and thus, needs to be finitely representable (either using a finite-memory or memoryless strategy). Moreover, in addition to providing restrictions on the input, we can provide qualifications for the other strategies reasoned about within the decision problem, such as the solution concept being searched for, as well as the deviations or counter-responses. With this in mind, we will use the term 'target strategy' to mean the strategy being 'searched for' in those decision problems which ask about the existence (or universality) of solution concepts. For example, the E-Nash problem asks if there exists some Nash equilibrium which models a provided specification — the target strategy here is the Nash equilibrium, and we will often be interested in restricting the associated search space.

Thus, we will adhere to the following convention throughout: when introducing a decision problem, we state it without providing any qualifications on the strategies (even if some finitely-representable strategy is required as input), but when providing and proving results about these decision problems, we will be careful to be clear about what assumptions we make of the strategies we work with; if the type of strategy is not explicitly stated, it should be assumed to be an arbitrary strategy.

Throughout this thesis, we will instantiate the above decision problems in a variety of contexts, with the focus being on adding a quantitative 'spin' to the questions.

Chapter 3

Themes on mean-payoff games

3.1 Introduction

In this chapter, we consider rational verification over multiplayer, mean-payoff games where system specifications are given by *Emerson-Lei conditions*. With this approach, the complexity of the main game-theoretic decision problems is considerably lower than in the case with temporal logic specifications. We consider the standard set of rational verification questions in the context of Nash equilibria and the core, as well as within a form of succinct representation of the usual concurrent game structures. We believe two of our contributions in this chapter to be particularly novel and significant:

1. As discussed in Chapter 2, Emerson-Lei conditions are endowed with more amenable computational properties than LTL. When we study rational verification decision problems in the setting of mean-payoff games, the addition of LTL specifications causes a complexity 'gap' between the **NP**-completeness of the non-emptiness problem [202] and the **PSPACE**-completeness of the E-Nash problem [110]. By using Emerson-Lei conditions in place of LTL specifications, we offer a way of reasoning qualitatively about mean-payoff games, without an overhead in complexity — the corresponding E-Nash problem remains **NP**-complete.

2. We extend the study of the core, a cooperative solution concept, within rational verification to mean-payoff games — a previously unexplored territory, with many new results and accompanying proof techniques.

We will proceed in the following way: after defining some chapter-specific preliminaries in Section 3.2, in Section 3.3 we study multiplayer mean-payoff games with Emerson-Lei conditions in the non-cooperative setting, and consider the natural decision problems relating to these games and their Nash equilibria. Following this, in Section 3.4 we shift our focus to the cooperative setting, and study the core in multiplayer mean-payoff games. Finally, in Section 3.5 we look at reactive module games [102] as a way of succinctly representing systems, and investigate how the use of this representation affects our complexity results.

3.2 Models, games, and specifications

Mean-payoff games

A two-player, mean-payoff game [87, 213], is defined by a tuple,

$$G = (V_1, V_2, v^0, E, w),$$

where,

- V_1 and V_2 are disjoint, finite, non-empty sets of states, controlled by player 1 and 2 respectively;
- $v^0 \in V_1$ is a designated start state;
- $E \subseteq (V_1 \times V_2) \cup (V_2 \times V_1)$ are the edges of the game;
- $w : E \to \mathbb{Z}$ is a weight function.

In a two-player, mean-payoff games, the players take turns choosing edges to traverse, each of which is weighted. Player 1 is trying to maximise the average of the weights encountered, whilst player 2 is trying to minimise it. Formally, the game begins in state $v^0 \in V_1$ and player 1 chooses an edge $(v^0, v^1) \in E \cap (V_1 \times V_2)$. Then player 2 chooses an edge $(v^1, v^2) \in E \cap (V_2 \times V_1)$ and this process repeats indefinitely. This induces a sequence of weights, $\vec{w} = w((v^0, v^1)), w((v^1, v^2)), ...$, and player 1 (respectively, 2) chooses edges to try and maximise (respectively, minimise) the *mean-payoff* of $\vec{w}$, denoted $\mathrm{mp}(\vec{w})$.

There are two keys facts about two-player, mean-payoff games that we will use without comment throughout. The first is that memoryless strategies suffice for both players to act optimally (*i.e.* achieve their maximum payoff) [87]. The second is that every game has a *value* (*i.e.* a payoff that player 1 can achieve regardless of what player 2 plays) and determining if a game's value is equal to v lies in **NP** $\cap$ **co-NP** [213]. In particular, given a game, determining its value can be seen as a problem that lies within **TFNP** [153].

Extending two-player, mean-payoff games to multiple players, a multiplayer mean-payoff game [202] is given by a tuple,

$$G = (M, \{w_i\}_{i \in \mathrm{Ag}}),$$

where,

- M is a concurrent game structure;
- For each agent $i \in \mathrm{Ag}$, $w_i : \mathrm{St} \to \mathbb{Z}$ is a weight function.

In a multiplayer mean-payoff game, a run of states $\pi = s^0 s^1 ...$ in M induces an infinite sequence of weights for each player i: $w_i(s^0) w_i(s^1) ....$ We denote this sequence by $w_i(\pi)$. Under a given path, π, a player's payoff is given by $\mathrm{mp}(w_i(\pi))$. For notational convenience, we will write $\mathrm{pay}_i(\pi)$ for $\mathrm{mp}(w_i(\pi))$. We can then define a preference relation over paths for each player as follows: we write $\pi \succeq_i \pi'$ and say that player i *prefers* π to π' if and only

if $\text{pay}_i(\pi) \geq \text{pay}_i(\pi')$. We also write $\pi >_i \pi'$ if $\pi \geq_i \pi'$ and not $\pi' \geq_i \pi$. Note that, since strategy profiles $\vec{\sigma}$ induce unique runs of states $\pi(\vec{\sigma})$, we can lift preference relations from runs to strategy profiles, writing $\vec{\sigma_1} \geq_i \vec{\sigma_2}$ as a shorthand for $\pi(\vec{\sigma_1}) \geq_i \pi(\vec{\sigma_2})$.

In what follows, we refer to multiplayer, mean-payoff games simply as mean-payoff games, and refer to two-player, mean-payoff games explicitly as such.

Emerson-Lei conditions in mean-payoff games

Let $\vec{\sigma}$ be some strategy profile. Then, $\vec{\sigma}$ induces some path, $\pi(\vec{\sigma})$, and given that Emerson-Lei conditions are defined on paths, we can talk about paths induced by strategies modelling them. But we are not interested in whether the paths induced by arbitrary strategies model a given Emerson-Lei condition — it is more natural in the context of multiplayer games to ask whether the paths induced by some or all of the equilibria of a game model a specification, both in the non-cooperative and in the cooperative contexts. In particular, we are interested in the Nash equilibrium and the core, and whether the paths induced by strategy profiles that are members of these solution concepts model a specification. To further illustrate the ideas presented so far, we offer the following example:

Example 1. Suppose we have four delivery robots in a warehouse (depicted in Figure 3.1; the four robots correspond to the four coloured triangles), who want to pick up parcels at the pickup points (labelled by the bold **P**s) and drop them off at the delivery points (labelled by the bold **D**s). If a robot is not holding a parcel, and goes to a pickup point, it automatically gets given one. If the robot has a parcel, and goes to the delivery point, then the parcel is taken from the robot, and it gains a payoff of 1. Finally, if two robots collide, by entering the same node at the same time, then they crash, and get a payoff of −999 at every future timestep.

Now, there are a number of Nash equilibria here (infinitely many, in fact), but it is easy to see that many of them exhibit undesirable properties. For instance, consider the strategy profile where red and pink go back and forth between the pickup and delivery

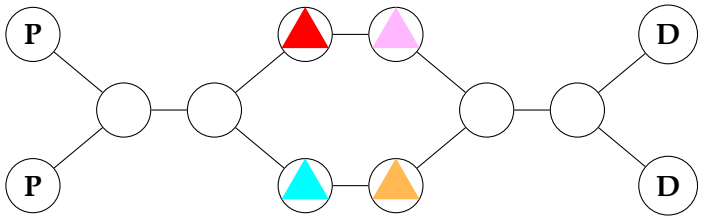

Figure 3.1: Robots manoeuvring in a warehouse.

points, and threaten to crash into, or deadlock, blue and yellow if they move from their starting positions. This is a Nash equilibrium, but it is clearly not Pareto optimal — the proposed strategy induces a socially undesirable outcome.

It is easy to identify a more egalitarian outcome — the one where all four robots visiting the pickup and delivery points infinitely often, waiting for the others to pass when they reach bottleneck points. If we call the set containing the two states where robot i visits a pickup point P_i and similarly label the set of delivery points D_i, we can express this requirement concisely with the following Emerson-Lei condition:

$$\bigwedge_{i \in [4]} \mathrm{Inf}(P_i) \wedge \mathrm{Inf}(D_i). \tag{3.1}$$

Thus, we can conclude that there exists some Nash equilibrium which models the above (generalised Büchi) specification. However, we just did this by inspection. In practice, we would like to ask this question in a more principled way. As such, we will spend the rest of this chapter exploring the natural decision problems associated with mean-payoff games with Emerson-Lei conditions.

Firstly, given a game, a strategy profile, and a specification given by an Emerson-Lei condition, we can ask if the strategy profile is an equilibrium whose induced path models the specification. Secondly, given a game and an Emerson-Lei condition, we can ask if the specification is modelled by the path/paths induced by some/every strategy profile in the set of equilibria of the game. Each of these problems can be phrased in the context of

a non-cooperative game or a cooperative game, depending on whether we let the set of equilibria be, respectively, the Nash equilibria or the core of the game. Formally, in the non-cooperative case, we have the following decision problems:

NASH-MEMBERSHIP:
Given: A mean-payoff game G, a strategy profile $\vec{\sigma}$, and an Emerson-Lei specification α.
Question: Is it the case that $\vec{\sigma} \in \text{NE}(G)$ and $\pi(\vec{\sigma}) \vDash \alpha$?

E-NASH:
Given: A mean-payoff game G and an Emerson-Lei specification α.
Question: Does there exist a $\vec{\sigma} \in \text{NE}(G)$ such that $\pi(\vec{\sigma}) \vDash \alpha$?

A natural dual to E-NASH is the A-NASH problem, which instead of asking if the specification holds in the path induced by *some* Nash equilibrium, asks if the specification holds in *all* equilibria:

A-NASH:
Given: A mean-payoff game G and an Emerson-Lei specification α.
Question: Is it the case that $\pi(\vec{\sigma}) \vDash \alpha$, for all $\vec{\sigma} \in \text{NE}(G)$?

As discussed previously, these decision problems were first studied in the context of iterated Boolean games [101], and are the 'flagship' decision problems of Rational Verification [102].

In the cooperative setting, the analogous decision problems are defined by substituting $\text{CORE}(G)$ for $\text{NE}(G)$. We refer to these problems as NASH-MEMBERSHIP, E-CORE, and A-CORE, respectively. These variants were first studied in the setting of LTL games [103].

Before we proceed to studying all these problems in detail, we note that even though some other types of games (for example, two-player, turn-based, zero-sum mean-payoff games, or parity games) can be solved only using memoryless strategies, this is not the

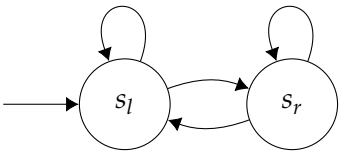

Figure 3.2: The game described in Proposition 2; it requires a infinite-memory Nash equilibrium to model an Emerson-Lei condition.

case in our setting; indeed, it turns out that infinite-memory strategies may be required to implement a Nash equilibrium which models a provided specification:

Proposition 2. *There exist games G and Emerson-Lei conditions α such that $\pi(\vec{\sigma}) \vDash \alpha$ for some Nash equilibrium $\vec{\sigma}$, but for which there exists no finite-memory Nash equilibrium $\vec{\sigma}$ such that $\pi(\vec{\sigma}) \vDash \alpha$. The statement also holds true for the core.*

Proof (construction suggested by Jean-François Raskin). Consider the game,

$$G = ((\mathrm{Ag}, \mathrm{St}, s^0, \{\mathrm{Ac}_i\}_{i \in \mathrm{Ag}}, \mathrm{tr}), \{w_i\}_{i \in \mathrm{Ag}}),$$

defined as follows:

- $\mathrm{Ag} = \{1\}$;
- $\mathrm{St} = \{s_l, s_r\}$;
- $\mathrm{Ac}_1 = \{\mathbf{S}, \mathbf{M}\}$;
- tr is defined in the following way: if player 1 plays **S**, then they remain in the state that they are currently in; if they instead play **M**, then they move to the other state.
- The sole weight function w_1 is as defined as $w_1(s_l) = 1$ and $w_1(s_r) = 0$.

An illustration of this game can be seen in Figure 3.2.

The game starts in the state s_l, and at each timestep player one can either choose to stay in the current state, or move to the other one. We let α denote the Emerson-Lei con-

dition $\mathrm{Inf}(s_r)$, and claim that there is no finite-memory Nash equilibrium that models this specification, but there is an infinite-memory one that does.

First, let σ_1 be any finite memory strategy for player 1, such that $\rho(\sigma_1) \vDash \alpha$. As σ_1 is finite memory, it induces a path consisting of a finite prefix, followed by an infinitely repeating suffix — that is, it can be written as $\rho(\sigma_1) = \pi_1 \pi_2^\omega$, where π_1 and π_2 are finite paths. As the satisfaction of an Emerson-Lei condition is prefix-independent, we see that s_r must occur in π_2 for $\rho(\sigma_1)$ to model α. In turn, as the mean-payoff of a sequence is also prefix-independent, we see that the mean-payoff of $\rho(\sigma_1)$ must be strictly less than 1 — it must be equal to the average of the weights encountered in π_2, which by the previous statement, includes 0. As player 1 can increase their payoff by deviating to the strategy which remains in s_l indefinitely, we may conclude that σ_1 is not a Nash equilibrium.

We can readily construct an infinite-memory strategy profile which is a Nash equilibrium which models α — consider the strategy σ_1' which visits s_l once, then s_r once, followed by visiting s_l twice, then s_r once, and continuing in this way, visiting s_l an increasing number of times each time it is reached, and only every visiting s_r once at a time. As s_r is visited infinitely often, we see that $\rho(\sigma_1') \vDash \alpha$. Moreover, whenever the game enters s_l, for some $n \in \mathbb{N}$, it will have visited s_l T_n times, where T_n is the n^{th} triangular number, and s_r n times. Thus, the average of the weights visited by this point must be equal to,

$$\frac{n^2 + n}{n^2 + 3n}.$$

It is easy to see that the limit of this sequence is 1, and thus the mean-payoff of $\rho(\sigma_1')$ is also 1. As player 1 achieves their highest possible payoff under this strategy, it is both a Nash equilibrium and a member of the core, and as established above, it also models the provided Emerson-Lei condition. □

3.3 Non-cooperative games

In the non-cooperative setting, Nash-Membership, E-Nash, and A-Nash are the relevant decision problems. In this section, we will show that Nash-Membership lies in P for memoryless strategies, while E-Nash is NP-complete and even remains NP-hard when restricted to memoryless strategies — thus, no worse than solving a multiplayer mean-payoff game [202]. Because A-Nash is the dual problem of E-Nash, it follows that A-Nash is co-NP-complete. In order to obtain some of these results, we also provide a semantic characterisation of the paths associated with strategy profiles in the set of Nash equilibria that satisfy a given Emerson-Lei condition. We will first study the Nash-Membership problem for memoryless strategies, and then investigate E-Nash, providing tight upper and lower bounds for both memoryless and arbitrary strategies. We start with memoryless strategies, as they are arguably the simplest, yet still computationally important, model of strategies one may want to consider for multi-agent systems.

Proposition 3. *Nash-Membership is in* **P** *when restricting the input to memoryless strategies.*

Proof. We verify that a given strategy profile is a Nash equilibrium in the following way. We begin by calculating the payoff of each player in the strategy profile by 'running' the strategy and keeping note of the game states. When we encounter the same game state twice, we can simply take the average of the weights encountered at the states between the two occurrences, and that will be the payoff for a given player. By the pigeonhole principle, it will take no more than $|\mathrm{St}|+1$ time steps to get to this point, and thus, this can be done in linear time.

Once we have each player's payoff, we can determine if they have any incentive to deviate in the following way — for each player, look at the graph induced by the strategy profile excluding them. Formally, the graph induced by a partial strategy profile $\vec{\sigma}_{-i}$, denoted by $\mathscr{G}[\vec{\sigma}_{-i}] = (V, E)$ is defined as follows: the set of nodes, V, simply consists of the set of states of the game, that is, $V = \mathrm{St}$ and the set of edges are simply the moves available

to player i — that is, we have $e = (s^1, s^2) \in E$ if and only if there exists some $\mathrm{ac}'_i \in \mathrm{Ac}_i$ such that $\mathrm{tr}(s^1, (\vec{\sigma}_{-i}(s^1), \mathrm{ac}'_i)) = s^2$.

We can use Karp's algorithm [127] to determine the maximum payoff that this player can achieve given the other players' strategies. If this payoff is higher than the payoff originally achieved, then this means player i can successfully deviate from the given strategy profile and be better off, and so it is not a Nash equilibrium. If we do this for each player, and the maximum payoff each player can achieve is less than or equal to their payoff in the given strategy profile, then we can conclude it is a Nash equilibrium, and moreover, we have determined this in polynomial time.

To determine if the strategy profile satisfies the specification, we run the strategy as before, and determine the periodic path which the game will end up in. This tells us which states will be finitely and infinitely visited, which in turn induces a valuation which will either model or not model the specification. Checking this can be done in polynomial time and thus, we can conclude that for memoryless strategies, Nash-Membership lies in **P**. □

The simplicity of the above algorithm may raise hopes that it might readily extend to finite-memory strategies. However, in this case, the configuration of the game is not just given by the current state — we must also account for the state that each of the player's strategies are in. Thus, we might have to visit at least $|\mathrm{St}| \cdot |Q|^{|\mathrm{Ag}|} + 1$ (where Q is the largest set of strategy states over the set of players) configurations until we discover a loop. Now whilst there is an exponential dependency on the number of players in the input to the problem, this bound on the number of configurations is not necessarily polynomial in the size of the input. More precisely, the size of the underlying concurrent game structure is $|\mathrm{St}| \cdot |\mathrm{Ac}|^{|\mathrm{Ag}|}$ and if $|Q|$ is larger that $|\mathrm{Ac}|$, the number of configurations will grow exponentially faster than the size of the input. Thus, we cannot use the above algorithm in the case of finite memory strategies to get a polynomial time upper bound. Given the structure of

this difficulty, we note that a parametric complexity analysis may yield fruitful insights here, but stop short of proceeding any further with this line of thinking.

We now consider the E-Nash problem. As shown by Ummels and Wojtczak [2011], the Non-Emptiness problem is **NP**-complete, and we dedicate the remainder of this section to matching this complexity bound under the introduction of Emerson-Lei specifications. Instead of providing the full **NP**-completeness result here, we start by showing that the problem is **NP**-complete for memoryless strategies, and delay the proof for the case of arbitrary strategies until we develop a useful characterisation of Nash equilibrium in the context of Emerson-Lei conditions.

We note that it is often the case that when we restrict the search space in a decision problem, it become more computationally difficult (for instance, linear programming can be done in polynomial time [114], but integer linear programming is **NP**-complete [126]). As such, one may wonder if the unrestricted version of E-Nash has sub-**NP** complexity, given its **NP**-completeness for memory strategies. However, this is not the case, and we will show that this problem is also **NP**-complete in general. For now, we have the following hardness result, obtained using a reduction from the Hamiltonian cycle problem [97, 126] — a similar argument can be found in [202].

Proposition 4. *E-Nash is* **NP***-hard when restricting the target strategy to memoryless strategies*[1]*, even for games with one player, and constant weights.*

Proof. Let $\mathscr{G}$ be a graph with $|V(\mathscr{G})| = n$. We form a mean-payoff game G by letting $\mathrm{Ag} = \{1\}$ and $\mathrm{St} = V(\mathscr{G})$. We pick the initial state arbitrary and label it s^0, and the actions for player 1 correspond to the edges of $\mathscr{G}$. That is, we have $\mathrm{Ac}_1 = E(\mathscr{G})$ and $\mathrm{tr}(u, e) = v$ if and only if $e = (u, v) \in E(\mathscr{G})$. We also fix an integer constant $k \in \mathbb{Z}$ and let $w_1(s) = k$ for all

[1] Note that we allow potential deviations to be arbitrary strategies; not just memoryless strategies.

$s \in \mathrm{St}$. Finally, let α be the following specification:

$$\alpha = \bigwedge_{s \in \mathrm{St}} \mathrm{Inf}(s).$$

We claim that $\mathscr{G}$ has a Hamiltonian cycle if and only if G has a memoryless Nash equilibrium $\vec{\sigma}$ such that $\pi(\vec{\sigma}) \vDash \alpha$.

First suppose that $\mathscr{G}$ has a Hamiltonian cycle, $\pi = v^0 v^1 \ldots v^{n-1}$. We define a memoryless strategy, σ_1, for player 1 by setting $\sigma_1(v^i) = v^{i+1}$, where the superscript is interpreted modulo n. As π is a Hamiltonian cycle, it visits every node, so σ_1 is a well-defined, total function. By definition, we see that $\pi(\sigma_1) = \pi$, so we have $\pi(\sigma) \vDash \alpha$. Additionally, as each state has the same payoff, this strategy is trivially a Nash equilibrium.

Now suppose that G has a memoryless Nash equilibrium σ_1 such that $\pi(\sigma_1) \vDash \alpha$. Let $\pi(\sigma_1) = \pi$. Since σ_1 is memoryless, π must be of the form $\pi = \pi[..i]\pi[i..j]^\omega$ for integers i and j. Without loss of generality, assume that i and j are the smallest integers such that this holds. Moreover, since $\pi \vDash \alpha$, the path π visits every state infinitely often, so we must have that $\pi[i..j]$ contains every state, and by memorylessness, it must be a cycle. Thus, by definition, $\pi[i..j]$ is a Hamiltonian cycle. □

Piecing Propositions 3 and 4 together establishes **NP**-completeness for mean-payoff games with Emerson-Lei conditions where the target strategy is memoryless[2]: one can non-deterministically guess a memoryless strategy for each player (which is simply a list of actions for each player, one for each state), and use Nash-Membership to verify that it is indeed a Nash equilibrium that models the specification.

To solve E-Nash in its full generality, we start by deriving the lower bound; in this setting, rather than using a reduction to the Hamiltonian cycle problem, we use an argument that appeals to SAT [76] instead:

Proposition 5. *E-Nash is* **NP***-hard, even for games with one player, and constant weights.*

[2]As above, we allow potential deviations to be arbitrary strategies here.

Proof. Let φ be a Boolean formula over the set of atomic propositions AP. We form a mean-payoff game G, where $\mathrm{Ag} = \{1\}$, $\mathrm{St} = \{s_p\}_{p \in \mathrm{AP}} \cup \{s_\perp\}$ and $\mathrm{Ac} = \{\mathrm{ac}_p\}_{p \in \mathrm{AP}} \cup \{\mathrm{ac}_\perp\}$. We set the start state, $s^0 \in \mathrm{St}$, arbitrarily, and let the transition function be defined as $\mathrm{tr}(s_p, \mathrm{ac}_q) = s_q$ for all states $s_p \in \mathrm{St}$. As before, we also fix an integer constant $k \in \mathbb{N}$ and set $w_1(s_p) = k$ for all $s_p \in \mathrm{St}$. Finally, we define an Emerson-Lei condition α_φ by replacing every occurrence of p in φ with $\mathrm{Inf}(s_p)$.

We now claim that φ is satisfiable if and only if G has a Nash equilibrium which models α_φ. First note that any strategy in G is a Nash equilibrium, as all strategies receive the same payoff. Thus it suffices to show that φ is satisfiable if and only if player 1 has a strategy in G whose induced run models α_φ.

To prove this claim, consider the following pair of functions $f : \mathscr{P}(\mathrm{AP}) \to \mathscr{P}(\mathrm{St})$, and $f^{-1} : \mathscr{P}(\mathrm{St}) \to \mathscr{P}(\mathrm{AP})$, defined as follows,

$$f(v) = \{s_p \mid p \in v\}, \quad f^{-1}(S) = \{p \mid s_p \in S\}.$$

It should be clear that these functions are inverses of each other, and are bijections. We now claim that $v \vDash \varphi$ if and only if there exists some run with permset $f(v)$ which models α_φ. First suppose that $\varphi = p$, where $p \in \mathrm{AP}$. By definition, $v \vDash p$ if and only if p is in v, and p is in v if and only if s_p is in $f(v)$, which in turn is true if and only if there exists some run with permset $f(v)$ which models $s_p = \alpha_\varphi$. Then, as Boolean formulae and Emerson-Lei conditions are structurally identical, with satisfaction for each being defined in an inductively identical way, and as the claim holds for atomic propositions, it holds for general formulae.

Thus, it suffices to show that there exists some run which models α_φ if and only if player 1 has a strategy whose induced run models α_φ. As the game graph of G is fully connected, this is apparent — every run can be achieved by some strategy of player 1, yielding our claim. □

With the lower bound for E-Nash established, we will turn our attention to the upper bound. To prove this, we need to develop an alternative characterisation of Nash equilibria. We begin by introducing the notion of the *punishment value* in a multiplayer mean-payoff game; cf., [106, 112]. The punishment value, $\mathrm{pun}_i(s)$, of a player i in a given state s can be thought of as the worst value the other players can impose on a player at a given state. Concretely, if we regard the game G as a two player, zero-sum game, where player i plays against the coalition $\mathrm{Ag} \setminus \{i\}$, then the punishment value for player i is the smallest mean-payoff value that the rest of players in Ag can inflict on i from a given state. Formally, given a player i and a state $s \in \mathrm{St}$, we define the punishment value, $\mathrm{pun}_i(s)$ against player i at state s, as follows:

$$\mathrm{pun}_i(s) = \min_{\vec{\sigma}_{-i} \in \Sigma_{-i}} \max_{\tau_i \in \Sigma_i} \mathrm{pay}_i(\pi((\vec{\sigma}_{-i}, \tau_i), s))$$

How efficiently can we calculate this value? As established by Ummels and Wojtczak [2011], we proceed in the following way: in a two player, zero-sum, mean-payoff game, memoryless strategies suffice to achieve the punishment value [87]. Thus, we can non-deterministically guess a pair of memoryless strategies for each player (one for the coalition punishing the player, and one for the player themselves), use Karp's algorithm [127] to find the maximum payoff for both the player and the coalition against their respective punishing strategies, and then verify that the two values coincide[3].

We can build on top of this definition of punishing values to define a notion of when the players are able to punish any player who deviates from an agreed upon run. Given a vector $\vec{z} \in \mathbb{R}^{|\mathrm{Ag}|}$, we say that a (state, action) pair, $(s, \vec{ac})$, is $\vec{z}$-*secure* if for all agents $i \in \mathrm{Ag}$

[3]Why do these values not coincide by definition? This is because the results of Ehrenfeucht and Mycielski [1979] are supplied for turn-based games, whilst here, we are in a concurrent setting, and a small amount of additional work is required to accordingly translate these results. We elide the details here, but refer the interested reader to Proposition 15 of Ummels and Wojtczak [2011]. The bottom line is that the technique described here is both necessary and sufficient for our purposes.

and for all $\mathrm{ac}'_i \in \mathrm{Ac}_i$, we have,

$$\mathrm{pun}_i(\mathrm{tr}(s, (\vec{\mathrm{ac}}_{-i}, \mathrm{ac}'_i))) \leq z_i.$$

Informally, a (state, action) pair is $\vec{z}$-secure if for every player, the remaining players have the ability to coordinate on the next timestep to ensure that the former player gets a payoff less than z_i. Intuitively, this can also be thought of as all the players having the power to 'punish' a player if they choose to unilaterally deviate from an agreed upon action profile. Indeed, we formalise this intuition in the following lemma, which we prove using techniques adapted from Gutierrez et al. [2019] and Ummels and Wojtczak [2011].

Lemma 1. *Let G be a game, $\{\vec{ac}[k]\}_{k\in\mathbb{N}}$ a sequence of actions, and π be the unique run of states induced by these actions. Then there is a Nash equilibrium, $\vec{\sigma} \in NE(G)$, such that $\pi = \pi(\vec{\sigma})$ if and only if there exists some $\vec{z} \in \mathbb{Q}^{Ag}$, with $z_i \in \{pun_i(s) \mid s \in St\}$[4], such that:*

- *for each k, we have that $(\pi[k], \vec{ac}[k])$ is $\vec{z}$-secure; and*
- *for all players $i \in Ag$, we have $z_i \leq pay_i(\pi)$.*

However, before we prove this, we will present an extended example to better illustrate the concepts defined above, as well as to motivate the steps taken in the remainder of this section.

Example 2. Consider a game G, with three players $\mathrm{Ag} = \{1, 2, 3\}$, seven states, $\mathrm{St} = \{s_0, s_1, s_2, s_3, s_p, s_{d_1}, s_{d_2}\}$, and three actions $\mathrm{Ac} = \{a, b\}$. Rather than writing down all $7 \times 2^3 = 56$ transitions explicitly, we define our transition function with the following rules:

- States s_p and s_{d_2} are sink states — once the game enters one of them, the game remains in the corresponding state for all future timesteps (16 transitions);

[4]There is a more constructive way of articulating this condition: the crucial value of $\vec{z}$ will be the player-wise maximum of their punishment values of the states within π as well as those states 'one deviation away'. However, given that we can 'absorb' this uncertainty of knowing which value of $\vec{z}$ to use into our **NP** guess in Theorem 1, we will provide the lemma as stated for overall ease of presentation.

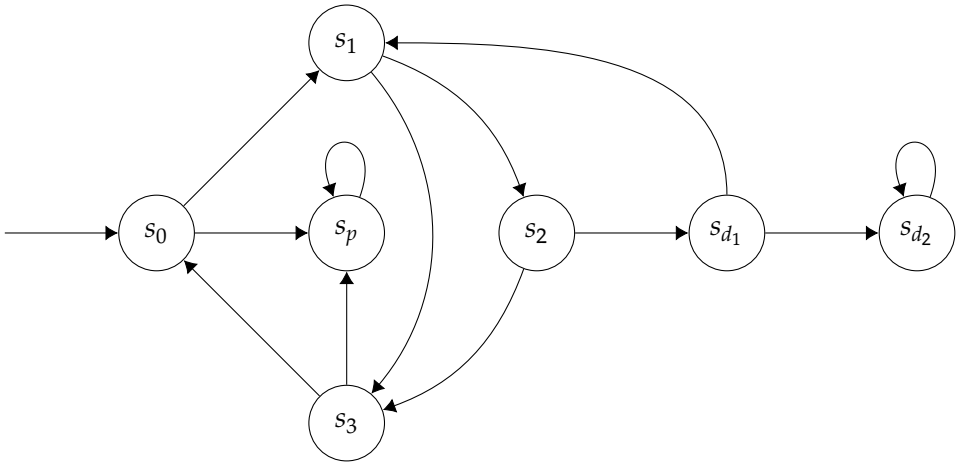

Figure 3.3: The game described in Example 2.

$w_i(s)$	1	2	3
s_0	1	1	1
s_1	1	1	1
s_2	1	1	1
s_3	1	1	1
s_p	–1	–1	–1
s_{d_1}	4	4	4
s_{d_2}	10	–1	–1

Table 3.1: The weight functions for the game in Example 2.

- In states s_0 or s_3, if at least two players play b, then the game moves to state s_p; otherwise, the game moves to state s_1 or s_0 respectively (16 transitions);
- In s_1, if at least two players play b, then the game moves to s_2; otherwise, the game moves to state s_3 (8 transitions);
- In s_2 and s_{d_1}, if, player 1 plays b, then the game moves to s_{d_1} and s_{d_2} respectively; otherwise it moves to s_3 and s_1 respectively (16 transitions).

Thus the game is as depicted in Figure 3.3. For each player, we define their weight functions as shown in Table 3.1.

Now, let us calculate the punishment value of each state for each player. It should be

$w_i(s)$	1	2	3
s_0	–1	–1	–1
s_1	–1	–1	–1
s_2	10	–1	–1
s_3	–1	–1	–1
s_p	–1	–1	–1
s_{d_1}	10	–1	–1
s_{d_2}	10	–1	–1

Table 3.2: The punishment values for each player in the game in Example 2.

apparent that in s_0 and s_3, each player can be punished by the other two both playing b, and forcing the game into s_p. As any run which ends up in s_p achieves the lowest possible payoff attainable in the game, we know that the punishment value, $\text{pun}_i(s_j)$, for all players in these two states is –1. For s_1, any two players can force the game into s_p by both of them playing a to get to s_3, and then playing b to enter s_p. Thus, the punishment value for each player in s_1 is –1. For s_2 and s_{d_2}, player 1 controls where the game goes — in particular they can force the game into s_{d_2} no matter what players 2 and 3 do. Thus, we see that the punishment values for players 2 and 3 are –1, whilst the punishment value of player 1 is 10. Finally, as sink states, the punishment values for each of the players in s_p and s_{d_2} are their respective weights. These punishment values are summarised in Table 3.2.

Having calculated these punishment values, we can now calculate the (state,action) pairs which are (–1, –1, –1)-secure. As the punishment values of s_1, s_0 and s_p are –1 for each player, it follows that for all action profiles $\vec{ac}$, $(s_0, \vec{ac})$, $(s_3, \vec{ac})$ and $(s_p, \vec{ac})$ are $(-1, -1, -1)$-secure. Now, in s_1, as two players need to play b to get the game into s_2, and the game moves to s_3 otherwise, we first note that $(s_1, (a, a, a))$ is $(-1, -1, -1)$-secure, as the game would still transition to s_3 under any unilateral deviation, and from there, the remaining two players can punish the third by moving the game to s_p. Additionally, $(s_1, (b, a, a))$ is $(-1, -1, -1)$-secure, as any deviation from player 1 would mean the game still moved to s_3, where they can be punished by players 2 and 3, and any deviation from either of the other two players lands the game in s_2, where player 1 can force the game into s_{d_2}, ensuring a

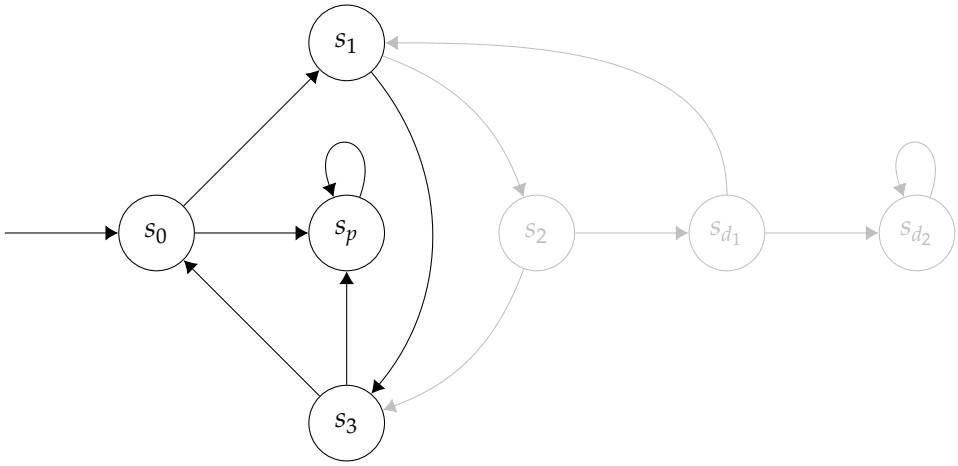

Figure 3.4: The game described in Example 2 with the non (–1, –1, –1)-secure transitions greyed out.

payoff for the former player of –1. However, for all other action profiles $\vec{ac}$, $(s_1, \vec{ac})$ is not (–1, –1, –1)-secure: one of players 2 and 3 must be playing b, so player 1 can play b to make the game move to s_2, where they can ensure a payoff of 10 for themselves by forcing the game into s_{d_1}, and then s_{d_2}. For the same reason, we also see that for all action profiles $\vec{ac}$, $(s_2, \vec{ac})$, $(s_{d_1}, \vec{ac})$ and $(s_{d_2}, \vec{ac})$ are not (–1, –1, –1)-secure.

If we then prune the game to only include those (state,action) pairs which are (–1, –1, –1)-secure, then the game appears as in Figure 3.4, where we have greyed out the non-(–1, –1, –1)-secure transitions, as well as any states that cannot be reached after any such transitions have been removed. We have kept the edge from s_1 to s_3 in the graph, with the understanding that only two of the associated transitions, (a, a, a) and (b, a, a), induce (–1, –1, –1)-secure (state, action) pairs.

This suggests a natural Nash equilibrium: the players coordinate by playing a in each state, which takes them cyclically around s_0, s_1, and s_3. If any one player deviates from this strategy, then the game remains in s_0, s_1 or s_3, and the remaining two players can punish the third by coordinating to move the game into s_p. Thus, as all the players get a payoff of 1, and any deviation would result in them getting a payoff of –1, we can conclude that this is a Nash equilibrium.

However, the keen-eyed reader may not that this outcome is not Pareto optimal. If the players were to coordinate and traverse s_1, s_2 and s_{d_1} cyclically, they would each achieve a higher payoff of 2. However, from s_{d_1}, player 1 can deviate to s_{d_2}, getting a payoff of 10 for themselves, and forcing a payoff of −1 on players 2 and 3, and we may conclude that this is *not* a Nash equilibrium. However, again, we could have concluded this by inspecting Figure 3.4. This highlights the strength of punishment values and the idea of $\vec{z}$-security, and intuitively captures the idea behind Lemma 1.

Having better motivated Lemma 1 (as well as results that follow it), we shall now offer its proof:

Proof of Lemma 1. First assume that we have some Nash equilibrium $\vec{\sigma} \in \mathrm{NE}(G)$ such that $\pi = \pi(\vec{\sigma})$. Suppose there does not exist any $z \in \mathbb{Q}^{\mathrm{Ag}}$ with the desired properties, and further suppose that for all $\vec{z} \in \mathbb{Q}^{\mathrm{Ag}}$, with $z_i \in \{\mathrm{pun}_i(s) \mid s \in \mathrm{St}\}$, there exists some player $i \in \mathrm{Ag}$ such that $z_i > \mathrm{mp}(w_i(\pi))$. But this is true for all $z_i \in \{\mathrm{pun}_i(s) \mid s \in \mathrm{St}\}$. So we have $\mathrm{pun}_i(s^0) > \mathrm{mp}(w_i(\pi))$. This means that if player i deviates to playing the memoryless strategy which ensures they achieve at least the punishment value, then they will get a better payoff, contradicting the fact that $\vec{\sigma}$ is a Nash equilibrium.

It must instead be the case that for all $\vec{z} \in \mathbb{Q}^{\mathrm{Ag}}$, with $z_i \in \{\mathrm{pun}_i(s) \mid s \in \mathrm{St}\}$, there is some time step k, such that $(\pi[k], \vec{\mathrm{ac}}[k])$ is not $\vec{z}$-secure. In particular, this means that there exists a player i and an action ac_i' such that,

$$\mathrm{pun}_i(\mathrm{tr}(\pi[k], (\vec{\mathrm{ac}}[k]_{-i}, \mathrm{ac}_i'))) > z_i.$$

But since this is true for all z_i, it is true for $z_i = \max\{\mathrm{pun}_i(s) \mid s \in \mathrm{St}\}$. And since we also have,

$$\mathrm{pun}_i(\mathrm{tr}(\pi[k], (\vec{\mathrm{ac}}[k]_{-i}, \mathrm{ac}_i'))) \in \{\mathrm{pun}_i(s) \mid s \in \mathrm{St}\},$$

this gives a contradiction. Thus, it follows that the first part of the statement is true.

Now assume that there exists some $\vec{z} \in \mathbb{Q}^{\mathrm{Ag}}$ with the properties as prescribed in the statement of the lemma. We define a strategy profile $\vec{\sigma}$ in the following way. Each player follows $\{\vec{\mathrm{ac}}[k]\}_{k\in\mathbb{N}}$ and π. If any player chooses to deviate from this run of actions, say at timestep k with an action ac_i' then the remaining players play the punishing strategy which causes player i to have a payoff of at most $\mathrm{pun}_i(\mathrm{tr}(\pi[k], \vec{\mathrm{ac}}[k]_{-i}, \mathrm{ac}_i'))$. Since we have

$$\mathrm{pun}_i(\mathrm{tr}(\pi[k], \vec{\mathrm{ac}}[k]_{-i}, \mathrm{ac}_i')) \leq z_i \leq \mathrm{pay}_i(\pi),$$

we see that no player has any incentive to deviate away from $\vec{\sigma}$ and so we have a Nash equilibrium with $\pi(\vec{\sigma}) = \pi$. □

With this lemma in mind, given a game G, and an Emerson-Lei condition α, along with a vector $\vec{z} \in \mathbb{R}^{|\mathrm{Ag}|}$, and a satisfying valuation $F \subseteq \mathrm{St}$ for the Emerson-Lei condition, we will define a graph which prunes the game graph of G to include only:

- Those transitions such that the associated (state, action) pair is $\vec{z}$-secure; and
- Those states in the valuation F.

Given Lemma 1, it should be clear that if such a graph has a path which visits every state infinitely often, and where the payoff of each player is greater than their corresponding component in $\vec{z}$, then this path models α and is sustained by some Nash equilibrium.

Formally, we define $\mathscr{G}[\vec{z}; F] = (V, E)$ as follows. We set $V = \mathrm{St}$ and include $(u, v) \in E$ if there exists some action profile $\vec{\mathrm{ac}}$ such that $v = \mathrm{tr}(u, \vec{\mathrm{ac}})$ with $(u, \vec{\mathrm{ac}})$ being $\vec{z}$-secure. Having done this, we then remove all states and edges not contained in F. Then given this definition and the preceding lemma, to determine if there exists a Nash equilibrium which satisfies an Emerson-Lei condition, α, we:

1. Calculate the punishment values, guess a vector $\vec{z_s} \in \mathrm{St}^{\mathrm{Ag}}$, as well a set of states, F, which satisfy α;

2. Let $z_i = \text{pun}_i(z_s)$ and form the graph $\mathscr{G}[\vec{z}; F]$;

3. Check if there is some path π in $\mathscr{G}[\vec{z}; F]$ with $z_i \leq \text{pay}_i(\pi)$ for each player i which visits every state infinitely often.

First, we note that if this graph is not strongly connected, then no path can visit every state infinitely often, and thus cannot satisfy α in G. So to determine if a path with the previously described properties exists, we check if it is strongly connected, and if so, appeal to one more piece of technical machinery, in the form of the following proposition:

Proposition 6. *Let $\mathscr{G}$ be a strongly connected graph, let $\{w_i\}_{i \in Ag}$ be a set of weight functions each with the signature $V(\mathscr{G}) \to \mathbb{Z}$, and let $\vec{z} \in \mathbb{Q}^{Ag}$. Then, we can determine if there is some path π with the properties,*

- *π visits every state infinitely often; and*
- *$z_i \leq pay_i(\pi)$ for each $i \in Ag$,*

in polynomial time.

Conceptually, Proposition 6 is similar to Theorem 18 of Ummels and Wojtczak [2011], but with a key difference — we need to do additional work to determine if there is a path that visits every state infinitely often.

For clarity of presentation, we will split the above proposition into two constituent lemmas. To do this, we begin by defining a system of linear inequalities. We then go on to show that there is a path π with the desired properties if and only if this system has a solution — one lemma for each direction. As the system of inequalities can be determined in polynomial time, this yields our result.

For a graph $\mathscr{G}$, a set of weight functions $\{w_i\}_{i \in \text{Ag}}$, and a vector $\vec{z} \in \mathbb{Q}^{\text{Ag}}$, we define a system of linear inequalities, $\ell(\mathscr{G}, \{w_i\}_{i \in \text{Ag}}, \vec{z})$ as follows: for each agent $i \in \text{Ag}$, and each edge $e \in E(\mathscr{G})$, introduce a variable $x_{i,e}$, along with the following inequalities:

i) $x_{i,e} \geq 0$ for each agent $i \in \mathrm{Ag}$ and for each edge $e \in E(\mathscr{G})$.

ii) $\sum_{e \in E(\mathscr{G})} x_{i,e} = 1$ for each agent $i \in \mathrm{Ag}$.

iii) $\sum_{e \in \mathrm{In}(v)} x_{i,e} = \sum_{e \in \mathrm{Out}(v)} x_{i,e}$ for all $i \in \mathrm{Ag}$ and for each $v \in V(\mathscr{G})$.

iv) $z_i \leq \sum_{e \in E(\mathscr{G})} x_{i,e} \cdot w_i(e)$ for all $i \in \mathrm{Ag}$.

v) $\sum_{e \in E(\mathscr{G})} x_{i,e} \cdot w_i(e) \leq \sum_{e \in E(\mathscr{G})} x_{j,e} \cdot w_i(e)$ for all $i, j \in \mathrm{Ag}$.

It is worth briefly discussing what this system is actually encoding. Roughly speaking, the set $\mathscr{C}_i = \{x_{i,e}\}_{e \in E}$ defines a cycle for each player that makes sure their payoff is greater than z_i. Each $x_{i,e}$ represents the proportion that a given edge is visited in the cycle. The idea is that we define a path by visiting each $\mathscr{C}_i$ an appropriate number of times, before travelling to the next cycle and visiting that repeatedly. Conversely, if there exists some path with the stated properties, it will also define a solution to the system of inequalities.

In what follows, for an edge $e = (u, v)$ and a weight function $w : V \to \mathbb{Z}$, we define $w(e) := w(u)$. We also extend weight functions to finite paths in the natural way, by summing along them.

Lemma 2. *Let $\mathscr{G}$ be a strongly connected graph, let $\{w_i\}_{i \in Ag}$ be a set of weight functions, let $\vec{z} \in \mathbb{Q}^{Ag}$. Furthermore, suppose there exists some path π such that $z_i \leq pay_i(\pi)$ for each $i \in Ag$, which visits every state infinitely often. Then $\ell(\mathscr{G}, \{w_i\}_{i \in Ag}, \vec{z})$ has a solution.*

Proof. First suppose there exists some path π with the stated properties. For each $n > 0$ and $e \in E(\mathscr{G})$, define $\lambda(n, e)$ to be the following quantity:

$$\lambda(n, e) = \frac{\#\{j < n \mid (\pi[j], \pi[j+1]) \in E(\mathscr{G})\}}{n}.$$

Informally, for a given edge e, $\lambda(n, e)$ gives us the proportion that e appears in the prefix $\pi[..n]$. Note that $0 \leq \lambda(n, e) \leq 1$ for all $e \in E(\mathscr{G})$ and all $n > 0$. Additionally, for each n, it is

easy to see we have,

$$\sum_{e \in E(\mathscr{G})} \lambda(n, e) = 1. \tag{3.2}$$

By definition, for each $i \in \mathrm{Ag}$ we have

$$\mathrm{pay}_i(\pi) = \liminf_{n \to \infty} \frac{1}{n} \sum_{k=0}^{n-1} w_i(\pi[k]).$$

By the definition of lim inf, there must exist some subsequence of the natural numbers, $\{n_t^i\}_{t \in \mathbb{N}}$, such that

$$\mathrm{pay}_i(\pi) = \lim_{t \to \infty} \frac{1}{n_t^i} \sum_{k=0}^{n_t^i - 1} w_i(\pi[k]).$$

With this defined, we introduce a sequence of numbers, $\{\varphi_t^i(e)\}_{t \in \mathbb{N}}$ for each edge $e \in E(\mathscr{G})$, by defining $\varphi_t^i(e) = \lambda(n_t^i, e)$. Now, as $\{\varphi_t^i(e)\}_{t \in \mathbb{N}}$ is a bounded sequence, by the Bolzano-Weierstrass theorem, there must be a convergent subsequence for each edge e, $\{\psi_t^i(e)\}_{t \in \mathbb{N}}$. We define $x_{i,e}^* = \lim_{t \to \infty} \psi_t^i(e)$ and claim that these form a solution to the system of inequalities.

Since $\varphi_t^i(e) = \lambda(n_t^i, e)$, it is clear that we have $0 \leq x_{i,e}^* \leq 1$. Thus, equation i) is satisfied. Additionally, by equation 3.2, we can deduce that for each t, and for every $i \in Ag$, we have

$$\sum_{e \in E(\mathscr{G})} \psi_t^i(e) = 1.$$

Taking limits, we can conclude that equation ii) is satisfied. To establish iii), by definition of λ, fix a $v \in V(\mathscr{G})$. In a path, if we enter a node, we must exit it. Thus, we have:

$$-1 \leq \sum_{e \in \mathrm{In}(v)} n \cdot \lambda(n, e) - \sum_{e \in \mathrm{Out}(v)} n \cdot \lambda(n, e) \leq 1,$$

implying,

$$-\frac{1}{n_t^i} \leq \sum_{e \in \text{In}(v)} \varphi_t^i(e) - \sum_{e \in \text{Out}(v)} \varphi_t^i(e) \leq \frac{1}{n_t^i}.$$

Taking the relevant subsequence and letting $t \to \infty$, we can deduce that equation iii) is satisfied. To establish iv), first note that for all $i \in \text{Ag}$, we have,

$$\begin{aligned}
\text{pay}_i(\pi) &= \liminf_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} w_i(\pi[k]) \\
&= \lim_{t\to\infty} \frac{1}{n_t^i} \sum_{k=0}^{n_t^i-1} w_i(\pi[k]) \\
&= \lim_{t\to\infty} \sum_{e \in E(\mathscr{G})} \lambda(n_t^i, e) \cdot w_i(e) \\
&= \lim_{t\to\infty} \sum_{e \in E(\mathscr{G})} \varphi_t^i(e) \cdot w_i(e) \\
&= \lim_{t\to\infty} \sum_{e \in E(\mathscr{G})} \psi_t^i(e) \cdot w_i(e) \\
&= \sum_{e \in E(\mathscr{G})} x_{i,e} \cdot w_i(e).
\end{aligned}$$

The equality between lines 4 and 5 is valid, as we have established that the limit exists in line 2. Thus, since we have $z_i \leq \text{pay}_i(\pi)$, by assumption, we can conclude that equation iv)

is valid. Similarly, note that for all $i, j \in \text{Ag}$, we have,

$$\begin{aligned}
\text{pay}_i(\pi) &= \liminf_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} w_i(\pi[k]) \\
&\leq \liminf_{t\to\infty} \frac{1}{n_t^j} \sum_{k=0}^{n_t^j-1} w_i(\pi[k]) \\
&= \liminf_{t\to\infty} \sum_{e\in E(\mathscr{G})} \lambda(n_t^j, e) \cdot w_i(e) \\
&= \liminf_{t\to\infty} \sum_{e\in E(\mathscr{G})} \varphi_t^j(e) \cdot w_i(e) \\
&\leq \lim_{t\to\infty} \sum_{e\in E(\mathscr{G})} \psi_t^j(e) \cdot w_i(e) \\
&= \sum_{e\in E(\mathscr{G})} x_{j,e} \cdot w_i(e).
\end{aligned}$$

This, together with equation iv), implies that equation v) is satisfied.

Thus, putting all this together, we can conclude that if there exists some path with the stated properties, then there also exists a solution to $\ell(\mathscr{G}, \{w_i\}_{i\in\text{Ag}}, \vec{z})$, the system of linear inequalities. □

We now show the other direction:

Lemma 3. *Let $\mathscr{G}$ be a strongly connected graph, let $\{w_i\}_{i\in Ag}$ be a set of weight functions, and let $\vec{z} \in \mathbb{Q}^{Ag}$. Suppose that $\ell(\mathscr{G}, \{w_i\}_{i\in Ag}, \vec{z})$ has a solution. Then there exists some path π such that $z_i \leq pay_i(\pi)$ for each $i \in Ag$, which visits every state infinitely often.*

Proof. Given that there is a solution to $\ell(\mathscr{G}, \{w_i\}_{i\in\text{Ag}}, \vec{z})$, there must exist a solution that consists of rational numbers. Thus, letting $x^*_{i,e}$ be a rational solution, we can write $x^*_{i,e} = p_{i,e}/q$, for some $p_{i,e} \in \mathbb{N}$ (we know that $p_{i,e} \geq 0$ by equation i)), and some appropriately chosen $q \in \mathbb{N}$.

Now, for each $i \in \mathrm{Ag}$, we form a multigraph, $\mathscr{G}_i = (V_i, E_i)$, which takes $\mathscr{G}$ and replaces each edge e by $p_{i,e}$ copies. Note that whilst $\mathscr{G}$ is strongly connected, some of the $p_{i,e}$ may be equal to 0, which would mean that $\mathscr{G}_i$ is disconnected.

Now, by equation iii), for each $v \in V$, we have

$$\sum_{e \in \mathrm{In}(v)} p_{i,e} = \sum_{e \in \mathrm{Out}(v)} p_{i,e}.$$

Thus, the in-degree of each vertex in V_i is equal to its out-degree. Thus, each strongly connected component of $\mathscr{G}_i$ contains an Eulerian cycle[5]. Interpreting each of these Eulerian cycles as paths in $\mathscr{G}$, we get a set of (not necessarily simple) cycles for each agent, $C_i^1, \ldots C_i^{m_i}$, where $m_i \leq |V|$.

Now for each agent $i \in \mathrm{Ag}$, and every $n \in \mathbb{N}$, we define a cycle $\mathscr{C}_i^n$ which starts at the first state of C_i^1, traces C_i^1 n times, takes the shortest path from the start state of C_i^1 to C_i^2, traces C_i^2 n times and repeats this process for all the cycles of agent i, until it has traced $C_i^{m_i}$ n times. From here, it takes the shortest path that visits every state, and then returns to the start state of C_i^1.

Let M_i be the largest of the absolute values of the weights of agent i. That is, $M_i = \max_{v \in V} |w_i(v)|$. Then we have for all $i, j \in \mathrm{Ag}$,

$$n \cdot \sum_{e \in E(\mathscr{G})} p_{j,e} \cdot w_i(e) - 2|V|^2 \cdot M_i \leq w_i(\mathscr{C}_j^n)$$

and,

$$|\mathscr{C}_j^n| \leq n \cdot \sum_{e \in E(\mathscr{G})} p_{j,e} + 2|V|^2.$$

[5]An Eulerian cycle is a path that starts and ends at the same node which visits every edge exactly once

Putting these together, we get

$$\begin{aligned}
\frac{w_i(\mathscr{C}_j^n)}{|\mathscr{C}_j^n|} &\geq \frac{n \cdot \sum_{e \in E(\mathscr{G})} p_{j,e} \cdot w_i(e) - 2|V|^2 \cdot M_i}{n \cdot \sum_{e \in E(\mathscr{G})} p_{j,e} + 2|V|^2} \\
&\geq \frac{n \cdot \sum_{e \in E(\mathscr{G})} p_{i,e} \cdot w_i(e) - 2|V|^2 \cdot M_i}{n \cdot \sum_{e \in E(\mathscr{G})} p_{j,e} + 2|V|^2} \quad \text{(by equation v))} \\
&= \frac{\sum_{e \in E(\mathscr{G})} x^*_{i,e} \cdot w_i(e) - \frac{2|V|^2 \cdot M_i}{n \cdot q}}{\sum_{e \in E(\mathscr{G})} x^*_{j,e} + \frac{2|V|^2}{n \cdot q}} \\
&\geq \frac{z_i - \frac{2|V|^2 \cdot M_i}{n \cdot q}}{1 + \frac{2|V|^2}{n \cdot q}} \quad \text{(by equation iv) and equation ii))}.
\end{aligned}$$

Taking limits, we see that

$$\lim_{n \to \infty} \frac{w_i(\mathscr{C}_j^n)}{|\mathscr{C}_j^n|} \geq z_i.$$

We are now ready to define a path π^n, which will form the basis for our path of interest, π. The path starts by visiting $\mathscr{C}^n_{n \bmod |\mathrm{Ag}|}$ $n!$ times, before taking the shortest path to the start of $\mathscr{C}^{n+1}_{n \bmod |\mathrm{Ag}|+1}$. We denote $\pi^{\cdots n}$ to be the finite prefix of π,

$$\pi^{\cdots n} = \pi^1 \pi^2 \ldots \pi^n,$$

and then form π by letting n go to infinity in $\pi^{\cdots n}$, so that,

$$\pi = \pi^1 \pi^2 \ldots \pi^n \ldots .$$

Now, by construction, π visits every state infinitely often. Thus, we just need to show that the payoff of π is greater than z_i. We start by calculating the payoff of a finite prefix, $\pi[\ldots n]$. This can be calculated via the following two bounds:

$$w_i(\pi[\ldots n]) \geq \sum_{j=1}^{n} j! \cdot w_i(\mathscr{C}^j_{j \bmod |\mathrm{Ag}|}) - n \cdot |V| \cdot M_i,$$

and,

$$|\pi^{\cdots n}| \leq \sum_{j=1}^{n} j! \cdot |\mathscr{C}_j^{\ j \bmod |\mathrm{Ag}|}|$$

To understand the quotient of these quantities, we first need to appeal to a small result, namely that we have,

$$\lim_{n\to\infty} \frac{1}{n!} \sum_{j=1}^{n-1} j! = 0.$$

This follows from the following inequality, which holds for all $n \in \mathbb{N}$ with $n \geq 3$, which can be proved by induction [202],

$$\frac{1}{n!} \sum_{j=1}^{n-1} j! < \frac{1}{n-2}.$$

Now, we note that we have both,

$$\lim_{n\to\infty} \frac{1}{n!} \left(\sum_{j=1}^{n} j! \cdot w_i(\mathscr{C}_j^{\ j \bmod |\mathrm{Ag}|}) - n \cdot |V| \cdot M_i \right) = \lim_{n\to\infty} w_i(\mathscr{C}_n^{\ n \bmod |\mathrm{Ag}|}),$$

and,

$$\lim_{n\to\infty} \frac{1}{n!} \sum_{j=1}^{n} j! \cdot |\mathscr{C}_j^{\ j \bmod |\mathrm{Ag}|}| = \lim_{n\to\infty} |\mathscr{C}_n^{\ n \bmod |\mathrm{Ag}|}|$$

Putting this together, we get:

$$\begin{aligned}
\mathrm{pay}_i(\pi) &= \liminf_{n\to\infty} \frac{w_i(\pi^{\cdots n})}{|\pi^{\cdots n}|} \\
&\geq \liminf_{n\to\infty} \frac{\sum_{j=1}^{n} j! \cdot w_i(\mathscr{C}_n^{\ n \bmod |\mathrm{Ag}|}) - n \cdot |\mathrm{Ag}| \cdot |V| \cdot M_i}{\sum_{j=1}^{n \cdot |\mathrm{Ag}|} j! \cdot |\mathscr{C}_n^{\ n \bmod |\mathrm{Ag}|}|} \\
&= \liminf_{n\to\infty} \frac{w_i(\mathscr{C}_n^{\ n \bmod |\mathrm{Ag}|})}{|\mathscr{C}_n^{\ n \bmod |\mathrm{Ag}|}|}
\end{aligned}$$

Now, the sequence $\frac{w_i(\mathscr{C}_n^{n \bmod |\mathrm{Ag}|})}{|\mathscr{C}_n^{n \bmod |\mathrm{Ag}|}|}$ is partitioned by the subsequences,

$$\left\{ \frac{w_i(\mathscr{C}_i^{n \cdot |\mathrm{Ag}| + i})}{|\mathscr{C}_i^{n \cdot |\mathrm{Ag}| + i}|} \right\}_{i \in \mathrm{Ag}}$$

Moreover, as established above, the limit of each of these subsequences is bounded below by z_i. Thus, we may conclude that we have,

$$\liminf_{n \to \infty} \frac{w_i(\mathscr{C}_n^{n \bmod |\mathrm{Ag}|})}{|\mathscr{C}_n^{n \bmod |\mathrm{Ag}|}|} \geq z_i,$$

finishing our proof. □

We can now combine these two results to obtain our proof:

Proof of Proposition 6. Given the statement of the proposition, we form the linear program $\ell(\mathscr{G}, \{w_i\}_{i \in \mathrm{Ag}}, \vec{z})$ and see if it has a solution. If it does, then a path with the desired properties exists, otherwise, it does not. As the linear program is of polynomial size, and as linear programs can be solved in polynomial time, we see that the overall algorithm can be done in polynomial time. By Lemma 3, we see that the algorithm is sound and then by Lemma 2, we see it is complete. □

From the propositions above, we can establish the complexity of E-Nash:

Theorem 1. *E-Nash is* **NP***-complete.*

Proof. For the lower bound, we have Proposition 5. For the upper bound, suppose we have an instance, (G, α), of the problem. Then we proceed as follows. We nondeterministically guess pairs of punishing strategy profiles, $(\zeta_i, \vec{\zeta}_{-i})$ for each player $i \in \mathrm{Ag}$, a state z_s for each player, and a set of states F. From these, we can easily check that any run with the permset F satisfies the specification, and we can also use Karp's algorithm to compute the punishment values, $\mathrm{pun}_i(s)$, for each state $s \in \mathrm{St}$ and for each player $i \in \mathrm{Ag}$.

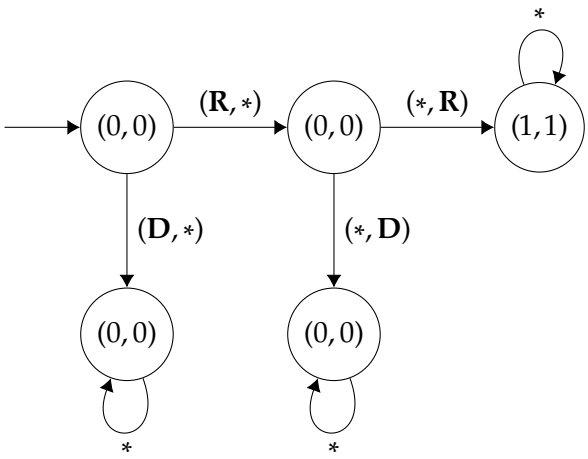

Figure 3.5: NE(G_1) $\not\subseteq$ CORE(G_1).

Setting $z_i = \text{pun}_i(z_s)$, we form the graph $\mathscr{G}[\vec{z}; F]$. We then invoke Lemma 1 together with Proposition 6, and accept or reject appropriately.

□

3.4 Cooperative games

We now move on to investigate cooperative solution concepts, and in particular, the core. We start by studying the relationship between Nash equilibrium and the core in our context, establishing that they are indeed different:

Proposition 7. *Let G be a game with $|Ag| = 1$. Then, $NE(G) = CORE(G)$. However, if $|Ag| \geq 2$, then there exist games G such that $NE(G) \not\subseteq CORE(G)$, and games G such that $CORE(G) \not\subseteq NE(G)$.*

Thus, the two concepts do not coincide beyond the one-player case. In fact, there are two-player games in which the set of Nash equilibria is empty, while the core is not, which demonstrates that the core is not a refinement of the Nash equilibrium. Nor does the other direction hold: the Nash equilibrium is not a refinement of the core.

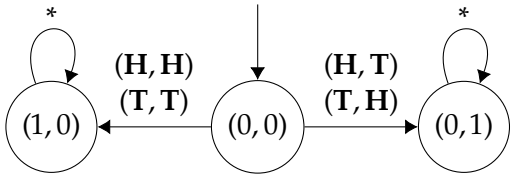

Figure 3.6: CORE(G_2) $\not\subseteq$ NE(G_2).

As we can see in the game in Figure 3.5, G_1 has a Nash equilibrium that is not in the core (and in which both players get a mean-payoff of 0 — cf., player 1 choosing **D** at the initial state while player 2 chooses **D** at the middle state), since the coalition containing both players has a beneficial deviation in which both players get a mean-payoff of 1. On the other hand, in the game in Figure 3.6, G_2 has a non-empty core (consider every possible memoryless strategy) while it has an empty set of Nash equilibria. Moreover, in both games, the detailed strategies can be implemented without memory. However, recall that in general, infinite-memory strategies may be required to implement all equilibria in the cooperative setting, as we showed in Proposition 2.

Another important (game-theoretic) question about cooperative games is whether they are guaranteed to have a non-empty core, a property that holds for games with LTL goals and specifications [103]. However, that is not the case for mean-payoff games, at least when we have three or more players:

Proposition 8. *In mean-payoff games, if* $|Ag| \leq 2$*, then the core is non-empty. For* $|Ag| > 2$*, there exist games with an empty core.*

Proof. If $|\mathrm{Ag}| = 1$, because of Proposition 7, the core coincides with the set of Nash equilibria in one-player games, which is always non-empty.

For two-player games, let $\vec{\sigma} = (\sigma_1, \sigma_2)$ be any strategy profile. If $\vec{\sigma}$ is not in the core, then either player 1, or player 2, or the coalition consisting of both players has a beneficial deviation. If the latter is true, then there is a strategy profile, $\vec{\sigma}' = (\sigma_1', \sigma_2')$ such that $\vec{\sigma}' >_i \vec{\sigma}$ for both $i \in \{1, 2\}$. We repeat this process until the coalition of both players does not have a

Ac	St
(H, H, H)	R
(H, H, T)	R
(H, T, H)	B
(H, T, T)	P
(T, H, H)	P
(T, H, T)	Y
(T, T, H)	B
(T, T, T)	Y

Table 3.3: The transition function for the game in Proposition 8.

beneficial deviation. This must eventually be the case as 1) each player's payoff is capped by their maximum weight and 2) by Theorem 4 of Brenguier and Raskin [2015], we see that the set of payoffs that a coalition can achieve is a closed set, so any limit point can be attained[6]. So there must come a point when they cannot beneficially deviate together. At this point, we must either be in the core, or either player 1 or player 2 has a beneficial deviation. If player $j \in \{1, 2\}$ has a beneficial deviation, σ_j, then any strategy profile (σ_j, σ_i), with $i \neq j$, that maximises player i's mean-payoff is in the core. Thus, for every two-player game, there exists some strategy profile that lies in the core.

However, for three-player mean-payoff games, in general, the core of a game may be empty. Consider the following three-player game G, where each player has two actions, **H**, **T**, and there are four states, P, R, B, Y. If the game is in any state other than P, then no matter what set of actions is taken, the game will remain in that state. Thus, we only specify the transitions for the state P, and provide them in Table 3.3. Additionally, the weight functions for each player are detailed in Table 3.4. With these components defined, the structure of this game is as depicted in Figure 3.7.

Note that strategies are characterised by the state that the game eventually ends up in. If the players stay in P forever, then they can all collectively change strategy to move to

[6]The aforementioned result of Brenguier and Raskin [2015] was proved in the context of turn-based games, but we can appeal to it here as our two-player game here is translated to a single player turn-based game; as one-player turn-based games behave like concurrent games, this logical step is valid.

$w_i(s)$	1	2	3
P	–1	–1	–1
R	2	1	0
B	0	2	1
Y	1	0	2

Table 3.4: The weight functions for the game in Proposition 8.

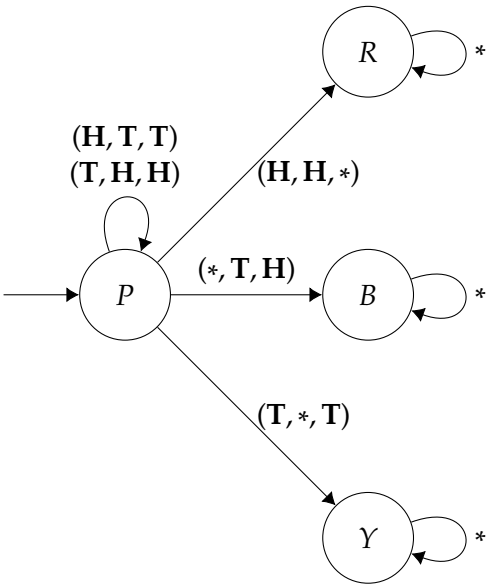

Figure 3.7: A game with an empty core.

one of R, B, Y, and each get a better payoff. Now, if the game ends up in R, then players 2 and 3 can deviate by playing $(\mathbf{T}, \mathbf{H})$, and no matter what player 1 plays, they will be in state B, and will be better off. But similarly, if the game is in B, then players 1 and 3 can deviate by playing $(\mathbf{T}, \mathbf{T})$ to enter state Y, in which they both will be better off, regardless of what player 2 does. And finally, if in Y, then players 1 and 2 can deviate by playing $(\mathbf{H}, \mathbf{H})$ to enter R and will be better off regardless of what player 3 plays. Thus, no strategy profile lies in the core. □

We pause for a moment to reflect on beneficial deviations; from a computational perspective, there is an immediate concern here — given a potential beneficial deviation, how can we verify that it is preferable to the status quo under *all* possible counter-responses?

Given that strategies can be arbitrary mathematical functions, how can we reason about that universal quantification effectively? We partially tackle this concern in the following lemma, by showing that for finite-memory strategies, we can restrict our attention to memoryless strategies when thinking about potential counter-responses to players' deviations:

Lemma 4. *Let G be a game, $C \subseteq Ag$ be a coalition and $\vec{\sigma}$ be a strategy profile. Further suppose that $\vec{\sigma}'_C$ is a finite-memory strategy vector such that for all memoryless strategy vectors $\vec{\sigma}'_{Ag\setminus C}$ and for all $i \in C$, we have,*

$$\pi(\vec{\sigma}'_C, \vec{\sigma}'_{Ag\setminus C}) \succ_i \pi(\vec{\sigma}).$$

Then, for all strategy vectors, $\vec{\sigma}'_{Ag\setminus C}$, not necessarily memoryless, and for all $i \in C$, we have,

$$\pi(\vec{\sigma}'_C, \vec{\sigma}'_{Ag\setminus C}) \succ_i \pi(\vec{\sigma}).$$

Before we prove this, we need to introduce an auxiliary concept of *two-player, zero-sum, multi-mean-payoff games* [207] (we will just call these multi-mean-payoff games moving forward). Informally, these are similar to two-player, zero-sum mean-payoff games, except player 1 has k weight functions associated with the edges, and they are trying to ensure the resulting k-vector of mean-payoffs is component-wise greater than a vector threshold. Formally, a multi-mean-payoff game is a tuple,

$$G = (V_1, V_2, v^0, E, w, z^k),$$

where,

- V_1, V_2 are finite, non-empty sets of states controlled by players 1 and 2 respectively;
- $V = V_1 \cup V_2$ is the state space;
- $v^0 \in V$ is a designated start state;

- $E \subseteq V \times V$ is a set of edges; and
- $w : E \to \mathbb{Z}^k$ a weight function, assigning to each edge a vector of weights.

The game is played by starting in the start state, $s^0 \in S_i$, and player i choosing an edge (s^0, s^1), and traversing it to the next state. From this new state, $s^1 \in S_j$, player j chooses an edge and so on, repeating this process forever. Paths are defined in the usual way and the payoff of a path π, $\text{pay}(\pi)$, is simply the vector $(\text{mp}(w_1(\pi)), \dots, \text{mp}(w_k(\pi)))$. Finally, $z^k \in \mathbb{Q}^k$ is a threshold vector and player 1 wins if the $\text{pay}_i(\pi) \geq z_i$ for all $i \in \{1, \dots, k\}$, and loses otherwise. An important question associated with these games is whether player 1 can force a win. As shown by Velner et al. [2015], this problem is **co**-**NP**-complete. Whilst we do not need to utilise this result right now, this sets us up to prove Lemma 4:

Proof of Lemma 4. Let $\vec{\sigma}_{\text{Ag}\setminus C}$ be an arbitrary strategy and let $i \in C$ be an arbitrary agent. Denote $\pi(\vec{\sigma})$ by π and $\pi(\vec{\sigma}'_C, \vec{\sigma}'_{\text{Ag}\setminus C})$ by π'. We aim to show that $\pi' \succ_i \pi$. Suppose instead it is the case that $\pi \succeq_i \pi'$. Thus, we have $\pi(\vec{\sigma}) \succeq_i \pi(\vec{\sigma}'_C, \vec{\sigma}'_{\text{Ag}\setminus C})$. Considering this as a two-player multi-mean-payoff game, where player 1's strategy is fixed and encoded into the game structure (*i.e.* player 1 follows $\vec{\sigma}'_C$, but has no say in the matter), and the payoff threshold is $\text{mp}(\pi(\vec{\sigma}))$, then $\vec{\sigma}'_{\text{Ag}\setminus C}$ is a winning strategy for player 2 in this game. Now, by Kopczynski [2006] and Velner et al. [2015], if player 2 has a winning strategy, then they have a memoryless winning strategy. Thus, there is a memoryless strategy $\vec{\sigma}''_{\text{Ag}\setminus C}$ such that $\pi(\vec{\sigma}) \succeq_i \pi(\vec{\sigma}'_C, \vec{\sigma}''_{\text{Ag}\setminus C})$. But this contradicts the assumptions of the lemma, and thus we must have $\pi' \succ_i \pi$[7]. □

We note that Lemma 4 shows that we may restrict our attention to memoryless counter-responses only when considering finite-memory deviations; we do not know if this result also holds for when the deviations are arbitrary strategies, and given the structure of our proof, we believe that if true, deriving such a result would require a different techniques.

[7] Velner et al. [2015] used a winning condition relating to whether a player's payoff is greater than or equal to a given vector. One can adapt this argument to show it is also true for strict inequalities.

We now look at some complexity bounds for mean-payoff games in the cooperative setting. We begin with one concerning beneficial deviations:

∃-Beneficial-Deviation (∃-Ben-Dev):

Given: A mean-payoff game G and a strategy profile $\vec{\sigma}$.

Question: Is there $C \subseteq \mathrm{Ag}$ and $\vec{\sigma}'_C \in \Sigma_C$ such that for all $\vec{\sigma}'_{\mathrm{Ag}\setminus C} \in \Sigma_{\mathrm{Ag}\setminus C}$ and for all $i \in C$, we have:

$$\pi(\vec{\sigma}'_C, \vec{\sigma}'_{\mathrm{Ag}\setminus C}) \succ_i \pi(\vec{\sigma})?$$

Using this new problem, we can prove the following statement.

Proposition 9. *Let G be a game, $\vec{\sigma}$ a strategy profile, and α a specification. Then, $(G, \vec{\sigma}, \alpha) \in$ Core-Membership if and only if $(G, \vec{\sigma}) \notin$ ∃-Ben-Dev and $\pi(\vec{\sigma}) \vDash \alpha$.*

Proof. Proof follows directly from definitions. □

The above proposition characterises the Core-Membership problem for cooperative games in terms of beneficial deviations, and, in turn, provides a direct way to study its complexity. In the remainder of this section we concentrate on the case of memoryless strategies: as before, their simple structure allows for natural, direct methods of proof, yet they remain an important, widespread model. Moreover, as shown in the non-cooperative setting, results yielded here can be suggestive of results in the general case.

Theorem 2. *When all players are restricted to memoryless strategies, ∃-Ben-Dev is* **NP***-complete.*

Proof (proof of lower bound due to Julian Gutierrez). First correctly guess a deviating coalition C and a strategy profile $\vec{\sigma}'_C$ (which is just a lookup table for each state due to its memorylessness) for such a coalition of players. Then, use the following three-step algorithm. First, compute the mean-payoffs that players in C get on $\pi(\vec{\sigma})$, that is, a set of values $z^*_j = \mathrm{pay}_j(\pi(\vec{\sigma}))$ for every $j \in C$ — this can be done in polynomial time simply by 'running' the strategy profile $\vec{\sigma}$. Then compute the graph $\mathscr{G}[\vec{\sigma}'_C]$, which contains all possible

behaviours (*i.e.* strategy profiles) for $\mathrm{Ag} \setminus C$ with respect to $\vec{\sigma}$ — this construction is similar to the one used in the proof of Proposition 3, that is, the game when we fix $\vec{\sigma'}_C$, and can be done in polynomial time. Finally, we ask whether every path π in $\mathscr{G}[\vec{\sigma'}_C]$ satisfies $\mathrm{pay}_j(\pi) > z_j^*$, for every $j \in C$ — for this step, we can use Karp's algorithm to answer the question in polynomial time for every $j \in C$. If every path in $\mathscr{G}[\vec{\sigma'}_C]$ has this property, then we accept; otherwise, we reject.

For hardness, we use a small variation of the construction presented in Sistla and Clarke [1985]. Let $P = \{x_1, \ldots, x_n\}$ be a set of atomic propositions. From a Boolean formula $\varphi = \bigwedge_{1 \leq c \leq m} C_c$ (in conjunctive normal form) over P — where each $C_c = l_{c1} \vee l_{c2} \vee l_{c3}$, and each literal $l_{ck} = x_j$ or $\neg x_j$, with $1 \leq k \leq 3$, for some $1 \leq j \leq n$ — we construct $M = (\mathrm{Ag}, \mathrm{St}, s^0, (\mathrm{Ac}_i)_{i \in \mathrm{Ag}}, \mathrm{tr})$, an m-player concurrent game structure defined as follows:

- $\mathrm{Ag} = \{1, \ldots, m\}$;
- $\mathrm{St} = \{x_v \mid 1 \leq v \leq n\} \cup \{x'_v \mid 1 \leq v \leq n\} \cup \{y_0, y_n, y^0, y^*\}$;
- $s^0 = y^0$;
- $\mathrm{Ac}_i = \{t, f\}$, for every $i \in \mathrm{Ag}$, and $\mathrm{Ac} = \mathrm{Ac}_1 \times \cdots \times \mathrm{Ac}_m$; and
- For tr, refer to the figure below, such that $T = \{(t_1, \ldots, t_m)\}$ and $F = \mathrm{Ac} \setminus T$.

The concurrent game structure so generated is illustrated in Figure 3.8. With M at hand, we build a mean-payoff game using the following weight function:

- $w_i(x_v) = 1$ if x_v is a literal in C_i and $w_i(x_v) = 0$ otherwise, for all $i \in \mathrm{Ag}$ and $1 \leq v \leq n$;
- $w_i(x'_v) = 1$ if $\neg x_v$ is a literal in C_i and $w_i(x'_v) = 0$ otherwise, for all $i \in \mathrm{Ag}$ and $1 \leq v \leq n$; and
- $w_i(y_0) = w_i(y_n) = w_i(y^0) = w_i(y^*) = 0$, for all $i \in \mathrm{Ag}$.

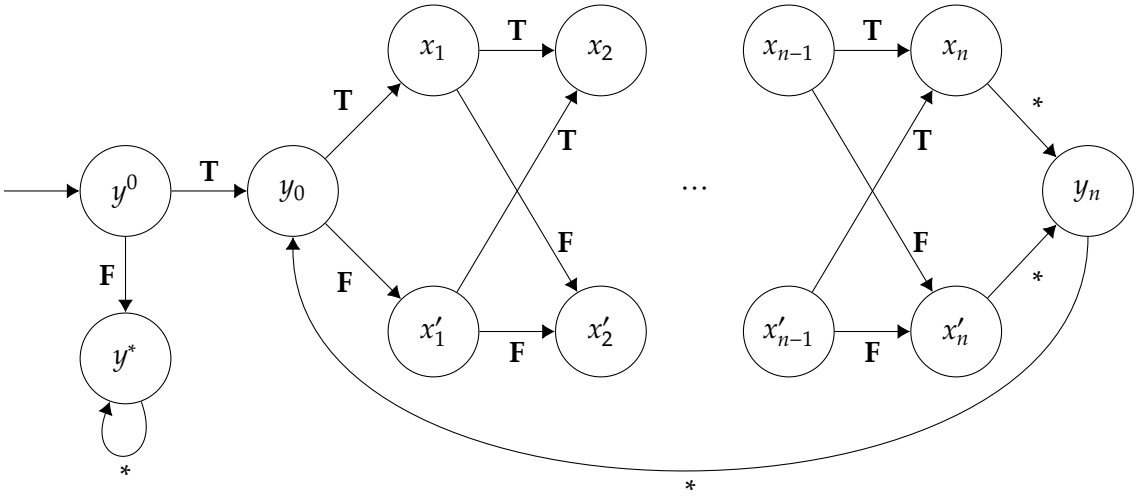

Figure 3.8: Concurrent game structure for the reduction from 3SAT in Theorem 2.

Then, we consider the game G over M and any strategy profile (in memoryless strategies) such that $\vec{\sigma}(s^0) = y^*$. For any of such strategy profiles the mean-payoff of every player is 0. However, if φ is satisfiable, then there is a path in M, from y_0 to y_n, such that in such a path, for every player, there is a state in which its payoff is not 0. Thus, the grand coalition Ag has an incentive to deviate since traversing that path infinitely often will give each player a mean-payoff strictly greater than 0. Observe two things. Firstly, that only if the grand coalition Ag agrees, the game can visit y_0 after y^0. Otherwise, the game will necessarily end up in y^* forever after. Secondly, because we are considering memoryless strategies, the path from y_0 to y_n followed at the beginning is the same path that will be followed thereafter, infinitely often. Then, we can conclude that there is a beneficial deviation (necessarily for Ag) if and only if φ is satisfiable, as otherwise at least one of the players in the game will not have an incentive to deviate (because its mean-payoff would continue to be 0). Then, formally, we can conclude that $(G, \sigma) \in \exists$-Ben-Dev if and only if φ is satisfiable. □

By Theorem 2, it follows that checking if no coalition of players has a beneficial devi-

ation with respect to a given strategy profile is a problem that lies in **co**-**NP**. More importantly, it also follows that Core-Membership is **co**-**NP**-complete.

Theorem 3. *When all players are restricted to memoryless strategies, Core-Membership is* **co**-**NP**-*complete.*

Proof. Recall that given a game G, a strategy profile $\vec{\sigma}$, and an Emerson-Lei condition α, we have $(G, \vec{\sigma}, \alpha) \in$ Core-Membership if and only if $(G, \vec{\sigma}) \notin \exists$-Ben-Dev and $\pi(\vec{\sigma}) \vDash \alpha$. Thus, we can solve Core-Membership simply by first checking $\pi(\vec{\sigma}) \vDash \alpha$, which can be done in polynomial time and we reject if that check fails. If $\pi(\vec{\sigma}) \vDash \alpha$, then we ask $(G, \vec{\sigma}) \in \exists$-Ben-Dev and accept if that check fails, and reject otherwise. Finally, since $\exists$-Ben-Dev is **NP**-hard, it follows from the above procedure that Core-Membership is **co**-**NP**-hard, which concludes the proof of the statement. □

$\exists$-Ben-Dev can also be used to solve E-Core in this case.

Theorem 4. *When all players are restricted to memoryless strategies, E-Core is in* Σ_2^P.

Proof. Given any instance (G, α), we guess a strategy profile $\vec{\sigma}$ and check that $\pi(\vec{\sigma}) \vDash \alpha$ and that $(G, \vec{\sigma}, \alpha)$ is not an instance of $\exists$-Ben-Dev. While the former can be done in polynomial time, the latter can be solved in co-NP using an oracle for $\exists$-Ben-Dev. Thus, we have a procedure that runs in $\mathbf{NP}^{\mathbf{co\text{-}NP}} = \mathbf{NP}^{\mathbf{NP}} = \Sigma_2^P$. □

By Theorem 4, it follows that when all players are restricted to memoryless strategies, A-Core is in Π_2^P and, more importantly, that even checking that the core has any memoryless solutions (but not necessarily that it is empty) is also in Σ_2^P. It is important to note that we have not managed to obtain corresponding lower bounds; however, given the 'layered' quality (in the sense of allowing counter-responses to any potential deviations) of the definitions of the cooperative concepts we have studied, relative to non-cooperative concepts and their complexities, our intuition is that these upper bounds are tight. With this proviso in mind, this result sharply contrasts with that for Nash equilibrium where

the same problem lies in **NP**. More importantly, the result also shows that the (complexity) dependence on the type of coalitional deviation is only weak, in the sense that different types of beneficial deviations may be considered within the same complexity class, as long as such deviations can be checked with an **NP** or **co-NP** oracle. For instance, Gutierrez, Kraus and Wooldridge [2019] defined other types of cooperative solution concepts, which differ from the one used in this thesis (known in the cooperative game theory literature as the α-core [169]) simply in the type of beneficial deviation under consideration.

We now lift the previous assumption that the space of strategies is memoryless; another concept introduced by Gutierrez, Kraus and Wooldridge [2019] is that of a 'fulfilled coalition', which informally characterises coalitions that have the strategic power (a joint strategy) to ensure a minimum given payoff *no matter what the other players in the game do* — an alternative interpretation in the context of LTL games (where this notion was introduced [103]) is that a coalition is fulfilled if it has a winning strategy against the counter-coalition. Generalising to our setting, from qualitative to quantitative payoffs, we introduce the notion of a *lower bound*: let $C \subseteq \mathrm{Ag}$ be a coalition in a game G and let $\vec{z}_C \in \mathbb{Q}^C$. We say that $\vec{z}_C = (z_1, \dots, z_i, \dots, z_{|C|})$ is a *lower bound* for C if there is a joint strategy $\vec{\sigma}_C$ for C such that for all strategies $\vec{\sigma}_{-C}$ for $\mathrm{Ag} \setminus C$, we have $\mathrm{pay}_i(\pi(\vec{\sigma}_C, \vec{\sigma}_{-C})) \geq z_i$, for every $i \in C$. Making a parallel with the above interpretation of fulfilled coalitions and winning strategies, in this context, a coalition has a lower bound if and only if they have a winning strategy against the counter-coalition in a certain multi-mean-payoff games; indeed, we will formalise this connection in Theorem 5. We also note that lower bounds can be seen as a generalisation of non-cooperative punishment values in the cooperative setting: with multiple players, in general there is no 'worst-case value' or outcome that can be forced upon them — some players will do better under one strategy and others will get higher payoffs under another. Thus, in the same way that punishment values represent the ability of one player being able to force their payoff to be at least a certain value when being adversarially targeted by the other players, lower bounds capture the idea of a coali-

tion being able to collectively force their payoffs to be above a certain multi-dimensional threshold.

Based on the definition above, we can prove the following lemma, which characterises the core in terms of paths on which payoffs can be ensured collectively, no matter any adversarial behaviour.

Lemma 5. *Let π be a path in G. There is $\vec{\sigma} \in CORE(G)$ such that $\pi = \pi(\vec{\sigma})$ if and only if for every coalition $C \subseteq Ag$ and lower bound $\vec{z}_C \in \mathbb{Q}^C$ for C, there is some $i \in C$ such that $z_i \leq pay_i(\pi)$.*

Proof (proof due to Julian Gutierrez). To show the left-to-right direction, suppose that there exists a member of the core $\vec{\sigma} \in \text{CORE}(G)$ with $\pi = \pi(\vec{\sigma})$ and suppose further that there is some coalition $C \subseteq \text{Ag}$ and lower bound $\vec{z}_C \in \mathbb{Q}^C$ for C, such that for every $i \in C$ we have $z_i > \text{pay}_i(\pi)$. Because $\vec{z}_C$ is a lower bound for C, and $z_i > \text{pay}_i(\pi)$, for every $i \in C$, then there is a joint strategy $\vec{\sigma}_C$ for C such that for all strategies $\vec{\sigma}_{-C}$ for $\text{Ag} \setminus C$, we have $\text{pay}_i(\pi(\vec{\sigma}_C, \vec{\sigma}_{-C})) \geq z_i > \text{pay}_i(\pi)$, for every $i \in C$. Then, it follows that $(G, \vec{\sigma}) \in \exists\text{-Ben-Dev}$, which further implies that $\vec{\sigma}$ cannot be in the core of G — a contradiction to our initial hypothesis.

For the right-to-left direction, suppose that there is π in G such that for every coalition $C \subseteq \text{Ag}$ and lower bound $\vec{z}_C \in \mathbb{Q}^C$ for C, there is $i \in C$ such that $z_i \leq \text{pay}_i(\pi)$. We then simply let $\vec{\sigma}$ be any strategy profile such that $\pi = \pi(\vec{\sigma})$. Now, let $C = \{j, \ldots, k\} \subseteq \text{Ag}$ be any coalition and $\vec{\sigma'}_C$ be any possible deviation of C from $\vec{\sigma}$. Either the vector $\vec{z}'_C$, where

$$\vec{z}'_C = (\text{pay}_j(\pi(\vec{\sigma}_{-C}, \vec{\sigma'}_C)), \ldots, \text{pay}_k(\pi(\vec{\sigma}_{-C}, \vec{\sigma'}_C))),$$

is a lower bound for C or it is not.

If we have the former, by hypothesis,, we know that there is $i \in C$ such that $\text{pay}_i(\pi(\vec{\sigma}_{-C}, \vec{\sigma'}_C)) \leq \text{pay}_i(\pi)$. Therefore, i will not have an incentive to deviate along with $C \setminus \{i\}$ from $\vec{\sigma}$, and as a consequence coalition C will not be able to beneficially deviate from $\vec{\sigma}$.

If, on the other hand, $\vec{z}'_C$ is not a lower bound for C, then, by the definition of lower bounds, we know that it is not the case that $\vec{\sigma}'_C$ is a joint strategy for C such that for all strategies $\vec{\sigma}'_{-C}$ for $\mathrm{Ag} \setminus C$, we have $\mathrm{pay}_i(\pi(\vec{\sigma}'_C, \vec{\sigma}'_{-C})) \geq \mathrm{pay}_i(\pi(\vec{\sigma}_{-C}, \vec{\sigma}'_C))$, for every $i \in C$. That is, there exists $i \in C$ and $\vec{\sigma}'_{-C}$ for $\mathrm{Ag} \setminus C$ such that $\mathrm{pay}_i(\pi(\vec{\sigma}'_C, \vec{\sigma}'_{-C})) < \mathrm{pay}_i(\pi(\vec{\sigma}_{-C}, \vec{\sigma}'_C))$. We will now choose $\vec{\sigma}'_{-C}$ so that, in addition, $\mathrm{pay}_i(\pi) \geq \mathrm{pay}_i(\pi(\vec{\sigma}'_C, \vec{\sigma}'_{-C}))$ for some i.

Now, let $\vec{z}''_C$ be a vector defined as

$$\vec{z''}_C = (\mathrm{pay}_j(\pi(\vec{\sigma}^j_{-C}, \vec{\sigma}'_C)), \ldots, \mathrm{pay}_k(\pi(\vec{\sigma}^k_{-C}, \vec{\sigma}'_C))),$$

where $\mathrm{pay}_i(\pi(\vec{\sigma}^i_{-C}, \vec{\sigma}'_C))$ is defined to be $\min_{\vec{\sigma}'_{-C} \in \Sigma_{-C}} \mathrm{pay}_i(\pi((\vec{\sigma}'_{-C}, \vec{\sigma}'_C)))$. That is, $\vec{\sigma}^i_{-C}$ is a strategy for $\mathrm{Ag} \setminus C$ which ensures the lowest mean-payoff for i assuming that C is playing the joint strategy $\vec{\sigma}'_C$. By construction $\vec{z}''_C$ is a lower bound for C— since each $z''_i = \mathrm{pay}_i(\pi(\vec{\sigma}^i_{-C}, \vec{\sigma}'_C))$ is the greatest mean-payoff value that i can ensure for itself when C is playing $\vec{\sigma}'_C$, no matter what coalition $\mathrm{Ag} \setminus C$ does — and therefore, by hypothesis we know that for some $i \in C$ we have $\mathrm{pay}_i(\pi(\vec{\sigma}^i_{-C}, \vec{\sigma}'_C)) \leq \mathrm{pay}_i(\pi)$. As a consequence, as before, i will not have an incentive to deviate along with $C \setminus \{i\}$ from $\vec{\sigma}$, and therefore coalition C will not be able to beneficially deviate from $\vec{\sigma}$. Because C and $\vec{\sigma}'_C$ where arbitrarily chosen, we conclude that $\vec{\sigma} \in \mathrm{CORE}(G)$, proving the right-to-left direction and finishing the proof. □

With this lemma in mind, we want to determine if a given vector, $\vec{z}_C$, is in fact a lower bound and importantly, how efficiently we can do this. That is, to understand the following decision problem:

> Lower-Bound:
>
> *Given*: A mean-payoff game G, a coalition $C \subseteq \mathrm{Ag}$, and a vector $\vec{z}_C \in \mathbb{Q}^{\mathrm{Ag}}$.
>
> *Question*: Is $\vec{z}_C$ is a lower bound for C in G?

Using the Multi-Mean-Payoff-Threshold decision problem introduced earlier, we can

prove the following proposition:

Theorem 5. *Lower-Bound is* **co**-**NP**-*complete.*

Proof. First, we show that Lower-Bound lies in **co**-**NP** by reducing it to Multi-Mean-Payoff-Threshold. Suppose we have an instance, $(G, C, \vec{z}_C)$, and we want to determine if it is in Lower-Bound. We can do this by forming the following two-player, multi-mean-payoff game, $G' = (V_1, V_2, v^0, E, w', z^k)$, where

- $V_1 = \mathrm{St}$;
- $V_2 = \mathrm{St} \times \mathrm{Ac}_C$;
- $v^0 = s^0$;
- $E = \{(s, (s, \mathrm{ac}_C)) \mid s \in \mathrm{St}, \mathrm{ac}_C \in \mathrm{Ac}_C\}$

  $\cup \{((s, \mathrm{ac}_C), \mathrm{tr}(s, (\mathrm{ac}_C, \mathrm{ac}_{\mathrm{Ag}\setminus C}))) \mid \mathrm{ac}_{\mathrm{Ag}\setminus C} \in \mathrm{Ac}_{\mathrm{Ag}\setminus C}\}$;
- $w' : E \to \mathbb{Z}^{|C|}$, with $w'_i(s, (s, \mathrm{ac}_C)) = w_i(s)$ and,

$$w'_i((s, \mathrm{ac}_C), \mathrm{tr}(s, (\mathrm{ac}_C, \mathrm{ac}_{\mathrm{Ag}\setminus C}))) = w_i(s); \text{ and}$$

- $z^{|C|} = \vec{z}_c$.

Informally, the two players of the game are C and $\mathrm{Ag} \setminus C$, the vector weight function is given by aggregating the weight functions of C and the threshold is $\vec{z}_C$. Now, if in this game, player 1 has a winning strategy, then there exists some strategy $\vec{\sigma}_C$ such that for all strategies of player 2, $\vec{\sigma}_{\mathrm{Ag}\setminus C}$, we have that $\pi(\vec{\sigma}_C, \vec{\sigma}_{\mathrm{Ag}\setminus C})$ is a winning path for player 1. But this means that $\mathrm{pay}_i(\pi(\vec{\sigma}_C, \vec{\sigma}_{\mathrm{Ag}\setminus C})) \geq z_i$ for all $i \in C$. But it is easy to verify that this implies that $\vec{z}_C$ is a lower bound for C in G. Conversely, if player 1 has no winning strategy, then for all strategies, $\vec{\sigma}_C$, there exists some strategy $\vec{\sigma}_{\mathrm{Ag}\setminus C}$ such that $\pi(\vec{\sigma}_C, \vec{\sigma}_{\mathrm{Ag}\setminus C})$ is not a winning path. This is turn implies that for some $j \in C$, we have that $\mathrm{pay}_j(\pi(\vec{\sigma}_C, \vec{\sigma}_{\mathrm{Ag}\setminus C})) < z_j$,

which means that $\vec{z}_C$ is not a lower bound for C in G. Also note that this construction can be performed in polynomial time, giving us the **co-NP** upper bound.

For the lower bound, we go the other way and reduce from Multi-Mean-Payoff-Threshold. Suppose we would like to determine if an instance G is in Multi-Mean-Payoff-Threshold. Then we form a mean-payoff game, G', with $k+1$ players, where the states of G' coincide exactly with the states of G. In this game, only the 1^{st} and $(k+1)^{\text{th}}$ player have any influence on the strategic nature of the game. If the game is in a state in V_1, player one can decide which state to move into next. Otherwise, if the game is in a state within V_2, then the $(k+1)^{\text{th}}$ player makes a move. Note we only allow moves that agree with moves allowed within G.

Now, in G', the first k players have weight functions corresponding to the k weight functions of player 1 in G. The last player can have any arbitrary weight function. With this machinery in place, we ask if z^k is a lower bound for $\{1, \dots, k\}$. In a similar manner of reasoning to the above, it is easy to verify that G is an instance of Multi-Mean-Payoff-Threshold if and only if z^k is a lower bound for $\{1, \dots, k\}$ in the constructed concurrent mean-payoff game. Moreover, this reduction can be done in polynomial time, and we can conclude that Lower-Bound is **co-NP**-complete. □

We have not presented any bounds for the complexity of E-Core in the general case. One possible reason that the upper bounds remain elusive to us is due to the fact that whilst in a multi-mean-payoff game, player 2 can act optimally with memoryless strategies, in general player 1 requires infinite memory to win [131, 207]. Given the close connection between the core in our concurrent, multi-agent setting and winning strategies in multi-mean-payoff games, this raises computational concerns for the E-Core problem. Additionally, Brenguier and Raskin [2015] studied the Pareto frontier of multi-mean-payoff games, and provide a way of constructing a representation of it, but this procedure has an exponential time dependency. The same paper also establishes Σ_2^P-completeness for the *polyhedron value problem*. Both of these problems appear to be intimately related to

the core, and we hope we might be able to use these results to gain more insight into the E-Core in the future.

With this having been said, we conclude this section by establishing a link between traditional non-transferable utility (NTU) games and our mean-payoff games — as NTU games are very well studied, and there is a wealth of results relating to core non-emptiness in this setting [40, 189, 191], we hope that some of these results could be utilised in order to understand the core of mean-payoff games.

Formally, an n-person game with NTU is a function,

$$V : \mathscr{P}\left(\mathrm{Ag}\right) \to \mathscr{P}\left(\mathbb{R}^{|\mathrm{Ag}|}\right),$$

such that,

- For all $C \subseteq \mathrm{Ag}$, $V(C)$ is a non-empty, proper, closed subset of $\mathbb{R}^{|\mathrm{Ag}|}$;
- For all $C \subseteq \mathrm{Ag}$, if we have $x \in V(C)$ and $y \in \mathbb{R}^{|\mathrm{Ag}|}$ such that $y_i \leq x_i$ for all $i \in C$, then we have $y \in V(C)$ (this property is sometimes called *downward closure*);
- We have that $V(\mathrm{Ag}) \setminus \bigcup_{i \in \mathrm{Ag}} \mathrm{int}\, V(\{i\})$ is non-empty and bounded.

We begin by giving a translation from mean-payoff games to NTU games. Let G be a mean-payoff game; then we define an NTU game, $G_{\mathrm{NTU}} = V : \mathscr{P}\left(\mathrm{Ag}\right) \to \mathscr{P}\left(\mathbb{R}^{|\mathrm{Ag}|}\right)$ as follows. If $C \subseteq \mathrm{Ag}$, then,

$$V(C) = \left\{\vec{z} \in \mathbb{R}^{|\mathrm{Ag}|} \mid \exists \vec{\sigma}_C \forall \vec{\sigma}_{\mathrm{Ag}\setminus C} \forall i \in C, \mathrm{pay}_i\left(\pi\left(\vec{\sigma}_C, \vec{\sigma}_{\mathrm{Ag}\setminus C}\right)\right) \geq z_i\right\}.$$

In words, $V(C)$ consists of the set of lower bounds that C can force[8]; by the link established in Theorem 5, it is also the set of values that C can (at least) ensure in an induced multi-mean-payoff game, and as such it is also exactly the downward closure of the Pareto

[8]Note that for an outcome $x \in V(C)$, the components x_i for $i \in \mathrm{Ag}\setminus C$ do not matter — they can be arbitrary real numbers.

curve of the values that can be attained by that coalition. To make a more general statement, we have that multiplayer mean-payoff games, their induced multi-mean-payoff games and their induced NTU games are highly interlinked — as such, results established for multi-mean-payoff games or NTU games readily translate into results about multiplayer mean-payoff games (a fact we have already taken advantage of).

We now show that this translation from mean-payoff games always correctly yields an NTU game.

Lemma 6. *Let G be a game, and let G_{NTU} be the NTU game associated with G. Then G_{NTU} is well-defined.*

Proof. We need to show that the three conditions in the definition of an NTU game hold for G_{NTU}.

For condition (1), we see that $V(C)$ is always non-empty by noting a coalition can always force an outcome where they achieve at least their worst possible payoff each (the vector made up of each player's lowest weight in the game). The fact that $V(C)$ is closed follows from Theorem 4 of Brenguier and Raskin [2015]. We also see that $V(C)$ is a proper subset of $\mathbb{R}^{|\text{Ag}|}$, as the members of C can do no better than achieve their maximum weights.

For condition (2), suppose we have $x \in V(C)$, and $y \in \mathbb{R}^{|\text{Ag}|}$ with $y_i \leq x_i$ for all $i \in C$. If $x \in V(C)$, then there exists some $\vec{\sigma}_C$, such that for all $\vec{\sigma}_{\text{Ag}\setminus C}$, we have $\text{pay}_i(\pi(\vec{\sigma}_C, \vec{\sigma}_{\text{Ag}\setminus C})) \geq x_i$ for all $i \in C$. But this in turn implies that $\text{pay}_i(\pi(\vec{\sigma}_C, \vec{\sigma}_{\text{Ag}\setminus C})) \geq y_i$ for all $i \in C$. Thus, by definition, we have $y \in V(C)$.

For condition (3), let p_j be the *punishment value* of the player j in the game G. Informally, the punishment value of a player j can be thought of as the worst payoff that the other players can inflict on that player. Alternatively, we can view the punishment value for player j as the best payoff they can guarantee themselves, no matter what the remaining players do — in this way, we can see that the punishment value is a maximal lower bound for a player.

Consider the vector $p \in \mathbb{R}^{|\text{Ag}|}$, where the j^{th} component of this vector is the punishment value for player j. Naturally, this vector lies in $V(\text{Ag})$. Additionally, we claim that is does not lie in $\text{int}\, V(\{i\})$ for any $i \in \text{Ag}$. For a contradiction, suppose there existed some $j \in \text{Ag}$ with $p \in \text{int}\, V(\{j\})$. So there exists some $\varepsilon > 0$, such that for all $0 \leq r < \varepsilon$, there exists some strategy, σ_j^r, such that for all counterstrategies, $\vec{\sigma}_{-j}$, we have $\text{pay}_j(\pi(\sigma_j^r, \vec{\sigma}_{-j})) \geq p_j + r$. But this implies player j can achieve a better payoff than their punishment value — a contradiction. Thus, we see that the set $V(\text{Ag}) \setminus \bigcup_{i \in \text{Ag}} \text{int}\ V(\{i\})$ is non-empty.

Finally, to see that $V(\text{Ag}) \setminus \bigcup_{i \in \text{Ag}} \text{int}\, V(\{i\})$ is bounded, we claim that it is contained in a closed ball of radius M, where M is defined to be

$$M = \max_{\substack{i \in \text{Ag} \\ s \in \text{St}}} |w_i(s)|.$$

We show that if $x \in V(\text{Ag})$, then we either have $x \in \bigcup_{i \in \text{Ag}} \text{int}\ V(\{i\})$ or $x \in \overline{B(0, M)}$, i.e., the closed ball of radius M, centred at the origin.

If $x \in V(\text{Ag})$, then by definition, we must have $x_i \leq M$ for all $i \in \text{Ag}$. Now, there are two possibilities: if we have $x_i \geq -M$ for all $i \in \text{Ag}$, then we have $x \in \overline{B(0, M)}$. So instead suppose there exists some $i \in \text{Ag}$ such that $x_i < -M$. In this case, letting ε be any positive number such that $x_i + \varepsilon \leq -M$, any strategy σ_i has the property that for all counter-strategies $\vec{\sigma}_{-i}$, we have $\text{pay}_i(\pi(\vec{\sigma}_{-i}, \sigma_i)) \geq x_i + \varepsilon$. Thus, we have $x \in \text{int}\, V(\{i\})$. This implies that,

$$V(\text{Ag}) \subseteq \bigcup_{i \in \text{Ag}} \text{int}\, V(\{i\}) \cup \overline{B(0, M)},$$

which in turn implies,

$$V(\text{Ag}) \setminus \bigcup_{i \in \text{Ag}} \text{int}\, V(\{i\}) \subseteq \overline{B(0, M)},$$

yielding the result. □

Given that we can translate mean-payoff games into well-defined NTU games, it is

natural to ask whether we can use traditional cooperative game theory in order to understand the core in our setting. Thus, we introduce the (classic) definition of the core for NTU games. In an NTU game, we say that an element $x \in \mathbb{R}^{|\mathrm{Ag}|}$ is in the core if $x \in V(\mathrm{Ag})$, and there exists no $C \subseteq \mathrm{Ag}$ and no $y \in V(C)$ such that $x_i < y_i$ for all $i \in C$. In the following result, we show that the core of a mean-payoff game, and the core of its corresponding NTU game are intimately related:

Lemma 7. *Let G be a mean-payoff game. Let G_{NTU} be the NTU game associated with G. Then the core of G is non-empty if and only if the core of G_{NTU} is non-empty.*

Proof. First suppose that G has a non-empty core. Thus, there exists some strategy profile $\vec{\sigma}$ such that for all coalitions C and for all strategy vectors $\vec{\sigma}'_C$, there exists some $\vec{\sigma}'_{\mathrm{Ag}\setminus C}$ such that $\mathrm{pay}_i(\pi(\vec{\sigma}'_C, \vec{\sigma}'_{\mathrm{Ag}\setminus C})) \leq \mathrm{pay}_i(\pi(\vec{\sigma}))$ for some $i \in C$. Let $x \in \mathbb{R}^{\mathrm{Ag}}$ be such that $x_i = \mathrm{pay}_i(\vec{\sigma})$ for all $i \in \mathrm{Ag}$. Then by definition, we have $x \in V(\mathrm{Ag})$. We claim that x is in the core of G_{NTU}. Suppose there is some $C \subseteq \mathrm{Ag}$ and a $y \in V(C)$ such that $x_i < y_i$ for all $i \in C$. Thus, there exists some $\vec{\sigma}'_C$ such that for all $\vec{\sigma}'_{\mathrm{Ag}\setminus C}$, such that $\mathrm{pay}_i(\pi(\vec{\sigma}'_C, \vec{\sigma}'_{\mathrm{Ag}\setminus C})) \geq y_i > x_i = \mathrm{pay}_i(\vec{\sigma})$ for all $i \in C$. But this implies that $\vec{\sigma}$ is not in the core of G, which is a contradiction. Thus, x is in the core of G_{NTU}.

Conversely, suppose that G_{NTU} has an empty core. Thus, there exists some $x \in \mathbb{R}^{|\mathrm{Ag}|}$ such that $x \in V(\mathrm{Ag})$, such that there exists no $C \subset \mathrm{Ag}$ and no $y \in V(C)$ with $x_i < y_i$ for all $i \in C$. Since $x \in V(\mathrm{Ag})$, there exists some strategy $\vec{\sigma}$ such that $\mathrm{pay}_i(\pi(\vec{\sigma})) \geq x_i$ for all $i \in \mathrm{Ag}$. We claim that $\vec{\sigma}$ is in the core of G. If it were not, then there would exist some coalition C, and some strategy vector $\vec{\sigma}'_C$ such that for all strategy vectors $\vec{\sigma}'_{\mathrm{Ag}\setminus C}$, we have $\pi(\vec{\sigma}'_C, \vec{\sigma}'_{\mathrm{Ag}\setminus C}) \succ_i \pi(\vec{\sigma})$ for all $i \in C$. We then define $y \in \mathbb{R}^{\mathrm{Ag}}$ by setting $y_i = \min_{\vec{\sigma}'_{\mathrm{Ag}\setminus C}} \mathrm{pay}_i(\pi(\vec{\sigma}'_C, \vec{\sigma}'_{\mathrm{Ag}\setminus C}))$ for $i \in C$ and setting $y_i = 0$ for $i \in \mathrm{Ag} \setminus C$. Then we have that $y \in V(C)$ by definition. Since for all $\vec{\sigma}'_{\mathrm{Ag}\setminus C}$, we have that $\pi(\vec{\sigma}'_C, \vec{\sigma}'_{\mathrm{Ag}\setminus C})$ is strictly preferred to $\pi(\vec{\sigma})$ by all players in C, we must have that $y_i > x_i$ for all $i \in C$. But this contradicts the fact that x is in the core of G_{NTU}. Thus, we must have that $\vec{\sigma}$ is in the core of G. □

As stated previously, we have been unable to determine the complexity of E-Core in the setting of mean-payoff games. However, given the above result, we suggest a route which may bear fruits in the future. Billera [1970], Bondareva [1963], Scarf [1967] and Shapley [1967, 1973] each reasoned about the core of cooperative games (in both the transferable utility and non-transferable utility settings) by appealing to the notion of a *balanced* set. Billera [1970] generalised this by introducing the notion of π-balancedness. Let $\pi = \{\pi_C \in \mathbb{R}^{|\mathrm{Ag}|} \mid C \subseteq \mathrm{Ag}\}$ be a collection of vectors such that,

- For all $C \subseteq \mathrm{Ag}$, we have $\pi_C \neq \mathbf{0}$;
- For all $C \subseteq \mathrm{Ag}$, and for all $i \notin C$, we have $\pi_{C,i} = 0$;
- For all $C \subseteq \mathrm{Ag}$, and for all $i \in C$, we have $\pi_{C,i} \geq 0$,

and let $\mathscr{C} \subseteq \mathscr{P}\left(\mathrm{Ag}\right)$ be a collection of coalitions. We say that $\mathscr{C}$ is π-balanced if there exist balancing weights, $\lambda_C > 0$, for each $C \subseteq \mathrm{Ag}$ such that,

$$\sum_{C \in \mathscr{C}} \lambda_C \pi_C = \pi_{\mathrm{Ag}}.$$

We then say that an NTU game, V, is π-balanced if whenever $\mathscr{C}$ is a π-balanced collection, we have,

$$\bigcap_{C \in \mathscr{C}} V(C) \subseteq V(\mathrm{Ag}).$$

Billera [1970] showed that if there exists some π such that V is π-balanced, then V has a non-empty core. The condition of π-balancedness translates readily over to the setting of mean-payoff games, and so we see that if such a game is π-balanced, then it has a non-empty core. This suggests a (sound, but not complete) algorithm for detecting if a mean-payoff game has a non-empty core; somehow guess a polynomially sized π, use a linear program to calculate the corresponding balancing weights, and then use an **co-NP** oracle to verify there exists no π-balanced collection such that $\bigcap_{C \in \mathscr{C}} V(C) \subseteq V(\mathrm{Ag})$.

Obviously, this is not a rigorous argument, but is suggestive of what a possible solution may look like.

Additionally, whilst π-balancedness is a sufficient condition for core non-emptiness, it is not necessary. However, Billera [1970] strengthened the condition of π-balancedness in the setting of convex-valued NTU games, to obtain a necessary and sufficient result. Given that in mean-payoff games, the outcomes that a coalition can achieve can be expressed as a union of convex sets, this approach seems promising. However, we have been unable to yield any results via this route, nor are we aware of any results in the literature for NTU games whose values are given by a union of convex regions or polytopes.

3.5 Weighted reactive module games

One problem with concurrent game structures as we have worked with them so far is that they are extremely verbose. The transition function, $\mathrm{tr} : \mathrm{St} \times \mathrm{Ac}_1 \times \cdots \times \mathrm{Ac}_{|\mathrm{Ag}|} \to \mathrm{St}$ is a total function, with size $|\mathrm{St}| \cdot |\mathrm{Ac}|^{|\mathrm{Ag}|}$. Thus, the size of the game scales exponentially with the number of the agents. In Example 1, the underlying concurrent game structure has a size of 429,981,696. If we are ever to have computational tools to support the decision problems described in this chapter, then such 'extensive' representations are not viable: we will require compact frameworks to represent games.

One natural framework we can use to induce succinctness is that of *reactive modules* [19]. Specifically, we modify the reactive module games of Gutierrez, Harrenstein and Wooldridge [2017] with weights on the guarded commands. We begin by walking through some preliminaries.

Reactive modules games do not use the full power of reactive modules, but instead use a subset of the reactive modules syntax, namely the *simple reactive modules language* (SRML) [119]. In SRML terms, agents are described by *modules*, which in turn consist of a set of variables controlled by the module, along with a set of *guarded commands*. The

basic idea is that under a given valuation of a set of variables, each module has a set of commands that are enabled for execution. Each module can then choose one enabled command, which reassigns the module's variables in a way which depends on the current valuation. This process repeats forever, and naturally, agents (represented by modules) will prefer some runs of valuations over others.

Formally, given a set of propositional variables Φ, a guarded command g is an expression of the form,

$$\varphi \rightsquigarrow x_1' := \psi_1; \ldots; x_k' := \psi_k,$$

where φ and each ψ_i are propositional formulae over Φ and each x_j also lies in Φ. We call φ the *guard* of g and denote it by guard(g), and we call the variables (the x_js) on the right-hand-side of g the *controlled variables* of g, denoted by ctr(g).

To define the size of a guarded command, we first define the size of a propositional formula, φ, denoted $|\varphi|$, to be the number of logical connectives it contains. Then the size of a guarded command g, written $|g|$, is given by $|\text{guard}(g)| + |\text{ctr}(g)|$.

With guarded commands in place, we can define the fundamental object of our computational model in this setting: given a set of propositional variables, Φ, a *simple reactive module*, is a tuple

$$m = (\Psi, I, U),$$

where,

- $\Psi \subseteq \Phi$ is a set of propositional variables;
- I is a set of *initialisation* guarded commands, where for all $g \in I$, we have $\text{guard}(g) = \top$ and $\text{ctr}(g) \subseteq \Psi$.
- U is a set of *update* guarded commands, where for all $g \in U$, guard(g) is a propositional formula over Φ and $\text{ctr}(g) \subseteq \Psi$.

After defining a simple reactive module m, we introduce an additional command, $g^{\mathbf{skip}}$, of the form,

$$\bigwedge_{g \in U} \neg \text{guard}(g) \rightsquigarrow \varnothing$$

with $\text{ctr}(g) = \Psi$. The empty set on the right-hand side of the guarded command means that no variables are changed. We introduce this extra guarded command so that at each stage, every module has *some* action they can take. However, this is not obligatory, and if we define a reactive module where we can prove at each step, there will be an available action, then introducing $g^{\mathbf{skip}}$ is not necessary. With this added, the *size* of a reactive module, $|m|$, is given by the sum of the sizes of its constituent guarded commands.

Given this, an SRML arena is a tuple

$$A = (\text{Ag}, \Phi, \{m_i\}_{i \in \text{Ag}}),$$

where,

- Ag is a finite, non-empty set of agents;
- Φ is a set of propositional variables;
- For each $i \in \text{Ag}$, m_i is a simple reactive module such that $\{\Psi(m_i)\}_{i \in \text{Ag}}$ is a partition for Φ.

We define the *size* of an arena to be sum of the sizes of the modules within it.

With this syntactic machinery in place, we are finally ready to describe the semantics of SRML arenas. We give a brief, high-level description here — for details, please refer to Gutierrez, Harrenstein and Wooldridge [2017] and van der Hoek, Lomuscio and Wooldridge [2006].

Let $v \subseteq \Phi$ be a valuation and let $i \in \text{Ag}$ be an agent. We let $\text{enabled}_i(v)$ denote the

guarded commands available to agent i under the valuation v. Formally, we have:

$$\text{enabled}_i(v) = \{g \in U_i \mid v \vDash \text{guard}(g)\}.$$

We then define $\text{enabled}(v) = \text{enabled}_1(v) \times \cdots \times \text{enabled}_{|\text{Ag}|}(v)$. Given that each U_i contains a command of the form $g_i^{\textbf{skip}}$ as described previously, we can see that each $\text{enabled}_i(v)$ is always non-empty. We also define $\text{exec}_i(g, v)$, which given a valuation v, is the valuation of Φ_i given by executing $g = \varphi \rightsquigarrow x_1' := \psi_1; \ldots; x_k' := \psi_k$. Specifically, we have:

$$\text{exec}_i(g, v) = (v_i \setminus \text{ctr}(g)) \cup \{x_j \mid v \vDash \psi_j\}.$$

Thus, upon executing g, agent i's variables are reassigned according to the valuations of the propositional formulae on its right-hand side. However, with multiple agents, there is some strategic interaction, and we wish to understand how the actions of all the agents affect the state of the system. As such, we define *joint guarded commands*, which are simply a selection of guarded commands, one for each agent. That is, we write $J = g_1 \times \cdots \times g_{|\text{Ag}|}$. Similarly to before, we set

$$\text{exec}(J, v) = \text{exec}_1(g_1, v) \cup \ldots \cup \text{exec}_{|\text{Ag}|}(g_{|\text{Ag}|}, v).$$

We are now ready to describe how arenas 'play out'. Initially, each agent i picks a guarded command $g_i^0 \in I_i$ and this sets each of their variables accordingly, inducing a valuation $v^0 = \text{exec}(J^0, \Phi)$. The agents then each pick a guarded command $g_i^1 \in \text{enabled}_i(v^0)$ and execute them, inducing another valuation, $v^1 = \text{exec}(J^1, v^0)$. They repeat this ad infinitum. As such, this induces a *path* of the game. However, unlike before, where we defined paths over actions and states of the game, here, we define paths over joint guarded commands, $\pi : \mathbb{N} \to (I_1 \cup U_1) \times \cdots \times (I_{|\text{Ag}|} \cup U_{|\text{Ag}|})$. Whilst, superficially, this may look like a departure from our previous convention, it is not; all the techniques we developed before

transfer readily to this new setting, so we take it for granted that there is a straightforward link between the two approaches and will not comment on it further.

We can now define *weighted reactive module games*. A weighted reactive module game (WRMG) is a tuple,

$$G = (A, \{w_i\}_{i \in \mathrm{Ag}}),$$

where,

- A is an SRML arena;
- For each $i \in \mathrm{Ag}$, $w_i : I(m_i(A)) \cup U(m_i(A)) \to \mathbb{Z}$ is a weight function.

That is, each module has an assigned weight function that maps commands to integers. As before, a player's payoff is given by the mean-payoff of the weights attached to a path. Note that we are effectively assigning weights to transitions, rather than to states as we did before. This is not a huge conceptual shift, and moving between the two representations (weights on states and weights on transitions) is a relatively straightforward transformation.

Given that the mean-payoff is prefix-independent, we could have defined our weight functions solely on the update commands (and not on initialisation command), or perhaps even defined our reactive modules without initialisation commands at all. However, for the sake of ascribing meaning to the game, and to make our upcoming proof techniques more general/'slick', we proceed as above, and define the weight functions over all commands available to the agents.

Finally, for our specifications, we use Emerson-Lei conditions over Φ. Thus, our specifications in this setting are Boolean combinations of atoms of the form $\mathrm{Inf}(p)$ with $p \in \Phi$. Moreover, since if p is a proposition, then $\neg p$ is also a proposition, we use the shorthand $\mathrm{Inf}(\neg p)$, with the obvious interpretation, in place of the previous $\mathrm{Inf}(\bar{F})$ notation. The semantics of these specifications are defined in the way described in Chapter 2.

Before considering the decision problems relating to WRMGs, let us now walk through an example to demonstrate their conciseness and utility of WRMGs. We do this by revisiting Example 1.

Example 3. In Example 1, the state of the game is entirely described by the position of each of the four robots, whether they are holding a parcel or not, and whether they have crashed. Thus, we define four reactive modules $m_1, \dots, m_4$ with $m_i = (\Phi_i, I_i, U_i)$ as follows — we set $\Phi_i = \{x_{i,1}, \dots, x_{i,12}, p_i, c_i\}$, where the x_i model which node the robot is in, numbered top-to-bottom, left-to-right with respect to Figure 3.1, p_i denotes if the robot is carrying a parcel or not, and c_i denotes if the robot has crashed or not. With this defined, we can define one initialisation command for each robot:

$$\top \rightsquigarrow x'_{i,1} := \bot; \dots x'_{i,n} := \top; \dots; x'_{i,12} := \bot; p'_i := \bot; c'_i := \bot \quad [0],$$

where n is appropriately set, given the starting position of the robot. Additionally, the [0] at the end of the guarded command denotes the weight rewarded for performing that command. Then for each agent i and edge (x_n, x_m) of the graph, we define a guarded command,

$$\neg c_i \wedge x_{i,n} \rightsquigarrow x'_{i,n} := \bot; x'_{i,m} := \top \quad [0].$$

We also model picking up and delivering a parcel, as well as crashing into another robot. We do this with the following commands:

$$\neg c_i \wedge \neg p_i \wedge (x_{i,1} \vee x_{i,2}) \rightsquigarrow p_i\prime := \top \quad [0],$$

$$\neg c_i \wedge p_i \wedge (x_{i,11} \vee x_{i,12}) \rightsquigarrow p_i\prime := \bot \quad [1],$$

$$\neg c_i \wedge x_{i,n} \wedge (x_{j,n} \vee x_{k,n} \vee x_{l_n}) \rightsquigarrow c'_i := \top \quad [-999]$$

$$c_i \rightsquigarrow c'_i := \top \quad [-999]$$

where i ranges over players; j, k and l range over the other players; and j ranges from 1 to 12. We also have the $g^{\mathbf{skip}}$ command from before, so the robot can stay still on a node for a time step.

We can also rewrite the specification of equation 3.1 in our new setting as follows:

$$\bigwedge_{i \in [4]} \mathrm{Inf}(p_i) \wedge \mathrm{Inf}(\neg p_i).$$

It is easy to see that this setup models the example from before, is *exponentially more concise*, requiring 52 guarded commands in total, and is natural to work with. Note that we could save even more space by encoding the robots positions in binary, at the expense of making our guarded commands slightly more complicated. Whilst this technique may be useful for larger systems, we give a unary encoding here for clarity. Moreover, our specification has not increased in size.

With WRMGs now adequately motivated, the main decision problem to consider then is the following:

WRMG-E-Nash:

Given: A WRMG G, and an Emerson-Lei condition α.

Question: Does there exist a $\vec{\sigma} \in \mathrm{NE}(G)$ such that $\pi(\vec{\sigma}) \vDash \alpha$?

Theorem 6. *The WRMG-E-Nash problem lies in* **NEXPTIME** *and is* **EXPTIME***-hard.*

Proof. The idea is to 'blow up' the simple reactive module arena into a concurrent game structure, then apply the same techniques as before. Explicitly, given a WRMG, $G = (\mathrm{Ag}, \Phi, \{m_i\}_{i \in \mathrm{Ag}}, \{w_i\}_{i \in \mathrm{Ag}})$, we form a graph $\mathscr{G} = (V, E)$ as follows. We set $V = \mathscr{P}\left(\Phi \cup \{p_{\mathrm{init}}\}\right)$ and then introduce an edge, $e = (v, w)$ if there exists some joint guarded update command J such that $\mathrm{exec}(J, v) = w$. We also introduce edges $(\{p_{\mathrm{init}}\}, v)$, if there exists some joint guarded initialisation command J such that $\mathrm{exec}(J, \varnothing) = v$. With $\mathscr{G}$ formed, we then construct a concurrent game structure, $G' = (\mathrm{Ag}', \mathrm{St}, s^0, (\mathrm{Ac}_i)_{i \in \mathrm{Ag}}, \mathrm{tr})$ in the natural way. We

set $\mathrm{Ag}' = \mathrm{Ag}$ and $\mathrm{St} = V$. We also set $s^0 = \{p_{\mathrm{init}}\}$. The actions for each player, Ac_i, consist of their respective guarded commands and the transition function corresponds to the edges of the graph. The weights of the guarded commands are attached to the weights of the transitions in the concurrent game structure [9].

We also need to transform the Emerson-Lei condition on the weighted reactive module games, α, to a specification on the concurrent game structure, α'. To do this, for each propositional variable p, define a subset $F_p \subseteq \mathrm{St}$ where $v \in F_p$ if $p \in v$. Then we simply replace every occurrence of $\mathrm{Inf}(p)$ in α with $\mathrm{Inf}(F_p)$.

It should be apparent that (G', α') exhibits the same behaviours and qualities as (G, α) and is exponential in size relative to (G, α). With this expansion complete, we can straightforwardly apply the **NP** algorithm of Proposition 1, immediately giving us the **NEXPTIME** upper bound.

To show hardness, we need to introduce another decision problem, Peek-G4 [197]. A Peek-G4 instance is a tuple (X_1, X_2, X_3, φ). Here, X_1 and X_2 are finite, disjoint, non-empty sets of propositional variables, $X_3 \subseteq X_1 \cup X_2$ gives the variables that are true in the initial state, and φ is a propositional formula over $X_1 \cup X_2$. The game is played by starting in the configuration given by X_3. The players then alternate, either choosing to toggle a variable they control, or skipping a go. If on their turn, a player can make φ true, then they win and the game is over. The decision problem associated with this game is to determine if agent 2 has a winning strategy for a give tuple (X_1, X_2, X_3, φ).

We reduce an instance of Peek-G4, (X_1, X_2, X_3, φ), to an instance of WRMG-E-Nash, $(\mathrm{Ag}, \Phi, \{m_i\}_{i \in \mathrm{Ag}}, \{w_i\}_{i \in \mathrm{Ag}})$. We set $\mathrm{Ag} = \{0, 1, 2\}$ and

$$\Phi = (X_1 \cup X_2) \cup \{\mathsf{turn_p}, \mathsf{turn_u}, \mathsf{win}_1, \mathsf{win}_2\},$$

[9] Whilst our analysis of E-Nash is couched in terms of weights on the states, in the proof of it, we modify the concurrent game structure so the weights exist on the transitions instead. Thus, this reduction does not need to introduce extra states to convert the weights on the transitions into weights on the states — we simply 'cut the corner' instead.

and introduce three reactive modules over Φ, m_0, m_1 and m_2. The modules m_1 and m_2 represent the two players, 1 and 2, whilst m_0 represents an umpire/officiator, who keeps track of whose turn it is and whether anyone has won the game yet.

The two players alternate, with the umpire enforcing this. That is, for m_0, we have $\Phi_0 = \{\mathsf{turn}_\mathrm{p}, \mathsf{turn}_\mathrm{u}, \mathsf{win}_1, \mathsf{win}_2\}$, with the following single initialisation command,

$$\top \rightsquigarrow \mathsf{turn}'_\mathrm{p} := \top; \mathsf{turn}'_\mathrm{u} := \bot; \mathsf{win}'_1 := \bot; \mathsf{win}'_2 := \bot.$$

The turn_p variable keeps track of which player's turn it is, and the turn_u variable determined if it is the umpire's turn. Finally, the two win variables keep track of whether anyone has won yet. The update commands for m_0 are defined as follows:

$$\begin{aligned}
\neg(\mathsf{win}_1 \vee \mathsf{win}_2) \wedge \neg\mathsf{turn}_\mathrm{u} &\rightsquigarrow \mathsf{turn}'_\mathrm{u} := \top, \\
\mathsf{turn}_\mathrm{u} \wedge \neg\varphi &\rightsquigarrow \mathsf{turn}'_\mathrm{u} := \bot; \mathsf{turn}'_\mathrm{p} := \neg\mathsf{turn}_\mathrm{p}, \\
\mathsf{turn}_\mathrm{u} \wedge \varphi \wedge \mathsf{turn}_\mathrm{p} &\rightsquigarrow \mathsf{win}'_1 := \top, \\
\mathsf{turn}_\mathrm{u} \wedge \varphi \wedge \neg\mathsf{turn}_\mathrm{p} &\rightsquigarrow \mathsf{win}'_2 := \top, \\
\mathsf{win}_1 \vee \mathsf{win}_2 &\rightsquigarrow \varnothing.
\end{aligned}$$

There is a lot going on in the above set of commands, so let us break it down: the first two commands say that play alternates between the players and the umpire — one of the players makes a move, then play goes to the umpire, who checks for wins, then play passes to the other player, and so on. The third and fourth commands say that if a player managed to satisfy φ on their last turn, then they have won, and the umpire can declare them the winner. The final command says that the umpire will just stay put and keep their variables constant once one player has won.

The two player modules are symmetric, so we define m_1, with m_2 being formed simil-

arly. For m_1, we have $\Phi_1 = X_1 \cup \{\mathsf{win}_2\}$, with a single initialisation command,

$$\top \rightsquigarrow x'_{i_1} := \top; \ldots; x'_{i_n} := \top,$$

where $\{x_{i_k}\}_{1 \leq k \leq n} = X_1 \cap X_3$. This simply says that player 1 must set their variables as dictated by X_3 Then for every variable $x_i \in X_1$, we introduce an update guarded command of the following form,

$$\neg(\mathsf{win}_1 \vee \mathsf{win}_2) \wedge \mathsf{turn}_\mathrm{p} \wedge \neg\mathsf{turn}_\mathrm{u} \rightsquigarrow x'_i := \neg x_i.$$

This says that if neither player has won yet, and it is player 1's turn to play, then they can toggle one of the variables they control. Additionally, we need to allow a player to be able not to change any of their variables, and we do this with the following guarded command,

$$\neg(\mathsf{win}_1 \vee \mathsf{win}_2) \rightsquigarrow \varnothing.$$

Note that player 1 can always play this command, even when it is not their turn. Finally, we need to introduce two more guarded commands that the players take when the game is over:

$$\mathsf{win}_1 \rightsquigarrow \varnothing$$
$$\mathsf{win}_2 \rightsquigarrow \varnothing.$$

If someone has won the game, then both players ignore the turn system and have the lone option of following exactly one of the above two commands forever.

With this defined, we need to introduce weight functions for each of the three players. For the umpire, we give a constant weight function (so they are ambivalent between all

outcomes) and for player 1 we give them a weight function w_1 such that,

$$w_1(\mathsf{win}_1 \rightsquigarrow \varnothing) = 1,$$
$$w_1(\mathsf{win}_2 \rightsquigarrow \varnothing) = -1,$$
$$w_1(g) = 0, \text{ for all other } g.$$

with player 2's weight function being defined in the dual way. Finally, the Emerson-Lei condition is simply $\alpha = \mathrm{Inf}(\mathsf{win}_2)$. We claim that player 2 has a winning strategy in Peek-G4 if and only if there exists a Nash equilibrium in the weighted reactive module game that models the given specification. Moreover, it is patently clear that this construction can be performed in polynomial time.

First suppose that player 2 has a winning strategy in Peek-G4. Then m_2 can play this strategy and m_1 can play any arbitrary strategy and eventually the game will end up in state win_2. From here, each player only has one guarded command they can play, meaning player 2 will get a payoff of 1 and player 1 will get a payoff of -1. This is a Nash equilibrium. No matter what m_1 plays, they are unable to force a win against player 2's winning strategy and so have no incentive to deviate. Additionally, player 2 is achieving their maximum payoff, and thus have no motivation to deviate. This strategy profile also models α.

Conversely, suppose there exists some Nash equilibrium which models α. This implies that the game eventually ends up in win_2, and so player 2 has a payoff of 1 and player 1 a payoff of -1 in this equilibrium. The fact that this is an equilibrium implies that no matter what player 1 plays, they cannot increase their payoff, i.e. they always lose against the strategy player 2 is playing. This implies that player 2 has a winning strategy in Peek-G4 and this concludes the proof. □

Chapter 4

Iterated allocation games

4.1 Introduction

As highlighted earlier, previous approaches to rational verification typically assume that agents are unrestricted in choosing their actions; however, in many real-world systems and situations, understanding resource usage is crucial in explaining the system's larger-scale behaviour — agents may not possess sufficient resources to achieve their individual goals, or they may want to optimise their behaviour and expend as few of the available resources as possible. In this chapter, we will consider games where agents have a limited number of resources they must allocate in order to achieve their respective goals.

There are a number of different 'types' of resources — for example, a resource can be thought of as a quantity that is consumed and produced over time, going up and down (such as money, or materials in manufacturing) or it can be thought of as a fixed amount of bandwidth or throughput of a system (such as electricity usage, data transfer speeds, or CPU load). de Nijs et al. [2017, 2021] refer to the former concept as being a *budget constraint*, and the latter as an *instantaneous constraint*. These are just two ways of thinking about resources, and the distinction here is relatively arbitrary — indeed, instantaneous constraints can be viewed as a special case of budget constraints, and other models of

resource-bounded systems may not fit neatly into this classification. However, for our purposes, it is a useful distinction, and we will use this terminology in this thesis to be explicit about how our formal model, and other models found elsewhere, should be interpreted.

In this chapter, we introduce *iterated allocation games*. These extend the one-shot *coalitional resource games* of Dunne et al. [2010] and Wooldridge and Dunne [2006] and can also be seen as a generalisation of *iterated Boolean games* [101]. Informally, an iterated allocation game consists of a set of agents, who over time allocate their instantaneous resources to 'activate' or 'enable' certain atomic propositions, with the hope that the induced run models their goal. However, because the goal of a player may depend on a propositional variable they do not have sole control over, strategic behaviour comes into play.

This chapter is organised as follows: in Section 4.2, we introduce our main objects of study, iterated allocation games, discuss the relevant related game theoretic concepts, and walk through an example. We then go on to define a number of decision problems pertaining to iterated allocation games in Section 4.3, establishing their complexities in turn, before considering a 'computationally easier' subset of LTL. In Section 4.4, we consider more resource-oriented decision problems (as opposed to the 'player'-oriented questions of Section 4.3), before refining our players' preferences, making them prefer computations in which resource expenditure is minimised, and consider a decision problem relating to this.

4.2 Iterated allocation games

In an iterated allocation game, we have a set of players, each of whom possesses a given quantity of resources. The resources we work with are instantaneous constraints: each agent has a fixed, constant amount of a resource at each time-step, they cannot produce more of the resource or stockpile it, and the quantity of each resource available to a player

returns to its base amount on each round of the game. Play proceeds in rounds — at each timestep, players allocate resources to different propositional variables. If a propositional variable has sufficient resources allocated to it across all the players, then it is considered to be true, otherwise, it is false. Doing this for each variable, this yields a valuation. This process then repeats, giving us an infinite sequence of valuations (which we call a run). Each player's goal is given by an LTL formula, which can be interpreted relative to this run — players prefer runs which model their goals over ones that do not, and are indifferent otherwise.

To formalise this idea, we first introduce *iterated allocation structures* — these model the game and how it is played, but do not give the agents goals. Formally, an *iterated allocation structure* (a IAS), is a tuple,

$$S = (\mathrm{Ag}, \mathrm{AP}, \mathrm{R}, \mathrm{en}, \mathrm{req}),$$

where,

- Ag, AP, and R are finite, non-empty sets of agents, propositional variables, and resources respectively;
- $\mathrm{en} : \mathrm{Ag} \to \mathbb{N}^R$ is an *endowment* function, which specifies the amount of each resource available to each player at each time step;
- $\mathrm{req} : \mathrm{AP} \to \mathbb{N}^R$ is a requirement function, which for each $p \in \mathrm{AP}$, can be understood as the amount of resources required to *satisfy* p.

The *size* of a IAS, S, denoted by $|S|$, is given by

$$|\mathrm{Ag}| + |\mathrm{AP}| + |R| + |\mathrm{Ag}| \cdot |R| \cdot \log_2 \max_{\substack{i \in \mathrm{Ag} \\ r \in R}} \mathrm{en}(i)_r + |\mathrm{AP}| \cdot |R| \cdot \log_2 \max_{\substack{p \in \mathrm{AP} \\ r \in R}} \mathrm{req}(p)_r.$$

Let us now detail how a IAS is played: informally, an *allocation* for a player i describes how that player assigns their resources to different atomic propositions on a given round of the game. Naturally, a player cannot allocate more resources than they are endowed with. Formally, an allocation for player i is a function,

$$\alpha_i : \mathrm{AP} \to \mathbb{N}^R,$$

which satisfies the following constraint:

$$\mathrm{en}(i) \geq \sum_{p \in \mathrm{AP}} \alpha_i(p).$$

Note that a player is not obliged to allocate all their resources on a round; they can 'hold back' resources if they desire.

Now let $\alpha : \mathrm{Ag} \to \mathrm{AP} \to \mathbb{N}^R$ be a vector of allocations, one for each player. Given a variable $p \in \mathrm{AP}$, we say that α *models* p, and write $\alpha \vDash p$, if we have,

$$\sum_{i \in \mathrm{Ag}} \alpha_i(p) \geq \mathrm{req}(p).$$

In other words, if the players collectively allocate enough resources to an atomic proposition, then it becomes 'true'. If they do not, then it is deemed to be 'false'. We use A_i to denote the set of allocations available to player i, and A to denote the Cartesian product, $A = A_1 \times \cdots \times A_n$. Finally, we note that an allocation $\alpha : \mathrm{Ag} \to \mathrm{AP} \to \mathbb{N}^R$ induces a unique valuation, $\mathrm{val}(\alpha)$, where, $\mathrm{val}(\alpha) = \{p \in \mathrm{AP} \mid \alpha \vDash p\}$.

Play proceeds as follows: at each time step, t, each agent chooses an allocation $\alpha_i[t]$. This forms a vector of allocations, $\alpha[t]$. This sequence of allocations then yields a run of valuations, $\mathrm{val}(\alpha[t]) \subseteq \mathrm{AP}$, where $t \in \mathbb{N}$. As a minor abuse of notation, we write $\vec{\alpha}$ for the sequence $\alpha[0]\alpha[1]$ ... and $\mathrm{val}(\vec{\alpha})$ for the run $\mathrm{val}(\alpha[0])\mathrm{val}(\alpha[1])$ ....

With game play detailed, we now discuss how each player chooses to play the game.

In this setting, a strategy for player i maps histories of gameplay (*i.e.* allocations provided thus far) to an allocation for that player; formally, it is a function,

$$\sigma_i : A^* \to A_i.$$

Here we note that a strategy profile σ induces a unique run in the obvious way — at the start of the game, player i chooses the allocation $\alpha_i[0] = \sigma_i(\varepsilon)$. This gives us a vector of allocations $\alpha[0]$, which in turn induces a valuation, $v^0 = \text{val}(\alpha[0])$. Continuing in this way, at timestep t of the game, player i will choose the allocation $\alpha_i[t] = \sigma_i(\alpha[0]\alpha[1] \ldots \alpha[t-1])$, yielding an allocation $\alpha[t]$, along with a valuation, $v^t = \text{val}(\alpha[t])$. This defines a run of valuations, $v^0 v^1 \ldots v^t \ldots$, which we denote as $\rho(\vec{\sigma})$.

As before, describing agent behaviours using arbitrary mathematical strategies is rather unwieldy, and again, it will be useful to talk about *finite-memory* strategies; these strategies are finitely representable, and therefore not only can they be reasoned about as inputs to decision problems, but also can be easily implemented in practice. Moreover, in iterated Boolean games (of which iterated allocation games are a generalisation), finite-memory strategies are sufficient to reason fully about player behaviour [101]: formally, for an LTL formula which is satisfiable over a iterated Boolean game, there is a finite memory strategy profile which models it.

Formally, for a given player $i \in \text{Ag}$, a finite memory strategy is a tuple,

$$\sigma_i = (Q_i, q_i^0, \delta_i, \tau_i),$$

where,

- Q_i is a finite, non-empty set of *strategy states*;
- $q_i^0 \in Q_i$ is a distinguished start state;
- $\delta_i : Q_i \times A \to Q_i$ is a transition function;

- $\tau_i : Q_i \to \text{AP} \to \mathbb{N}^R$ is a choice function, such that for all $q \in Q_i$, we have that $\tau_i(q)$ is a valid allocation for i.

The interpretation of a finite memory strategy is that a player begins in the strategy start state, q_i^0, and then chooses an allocation $\alpha_i[0] = \tau_i(q_i^0)$. Along with the other players' allocations (which may not necessarily be produced by finite memory strategies themselves), this yields a vector of allocations, $\alpha[0]$, which in turn induces a valuation $\text{val}(\alpha[0])$. The player then moves to a new strategy state, $q_i^1 = \delta_i(q_i^0, \alpha[0])$. This process is then repeated ad infinitum, inducing a run of valuations in the same way described above. Note that every finite memory strategy is indeed a valid strategy in the sense that they map sequences of valuations to allocations. However, hereafter, unless otherwise noted, we work with arbitrary strategies.

Now that we have detailed how these games are played, we are ready to introduce player goals into the fray. An iterated allocation game (IAG) is a tuple,

$$G = (S, \{\gamma_i\}_{i \in Ag}),$$

such that,

- S is an iterated allocation structure;
- For each agent $i \in \text{Ag}$, γ_i is an LTL formula called the *goal* of agent i.

The size of a IAG G, denoted $|G|$, is $|S| + \sum_{i \in \text{Ag}} |\gamma_i|$.

Suppose we have an infinite sequence of allocations, $\vec{\alpha}$, with an associated run, $\text{val}(\vec{\alpha})$. Then the LTL goal of each player, γ_i, can be interpreted relative to this run — if $\text{val}(\vec{\alpha}) \vDash \gamma_i$, we say that player i's goal is *satisfied* and call player i a winner of the game. If this is not the case, then we say that player i's goal is not satisfied, and that player i is a *loser* of the game. We say that a player i *prefers* a run $\vec{\alpha}$ to a run $\vec{\alpha}'$, and write $\vec{\alpha} \succeq \vec{\alpha}'$ if i is a winner in

$\vec{\alpha}$ (note the lack of dependence on $\vec{\alpha}'$). We also write $\vec{\alpha} > \vec{\alpha}'$ if $\vec{\alpha} \succeq \vec{\alpha}'$ but not $\vec{\alpha}' \succeq \vec{\alpha}$, and $\vec{\alpha} \sim \vec{\alpha}'$ if $\vec{\alpha} \succeq \vec{\alpha}'$ and $\vec{\alpha}' \succeq \vec{\alpha}$.

In a IAG, we would like to understand what sort of outcomes might emerge, under the assumption that players are acting rationally. To do this, we use the machinery of game theory, and analyse our games with respect to the Nash equilibrium and the core.

Before proceeding, we present an example to illustrate the concepts presented thus far:

Example 4. Suppose we have three servers, nicknamed Ceres, Pluto and Eris. Each server is equipped with a certain number of CPU cores and GPUs, and has a fixed amount of network bandwidth (measured in Mbps). Each server runs a set of core services (like systemd), and together, they have some pre-assigned functions, namely serving email and web pages. Additionally, a couple of graduate students are racing to get some last minute results for their paper, and submit some high-load algorithms to the job-scheduler. We want to know whether the servers can complete the grad students' work in time, subject to the pre-assigned constraints of the servers. To model this, we create a IAG as follows:

- We set $\mathrm{Ag} = \{\mathrm{C}, \mathrm{P}, \mathrm{E}\}$, representing the three servers;
- We set $R = \{\mathrm{CPU}, \mathrm{GPU}, \mathrm{Mbps}\}$, representing CPU cores, GPU cores and network bandwidth, respectively. We assume that each timestep in the IAG corresponds to a real-life second, and a unit of a resource should be interpreted as load on that resource over that period of time;
- We set $\mathrm{AP} = \{\mathrm{SD}_i \mid i \in \mathrm{Ag}\} \cup \{\mathrm{EMAIL}, \mathrm{WEB}\} \cup \{\mathrm{GA}_k \mid k \in [100]\}$, where SD_i represent the system services of server i, EMAIL and WEB respectively represent email and web pages being served, and GA_k represents the k^{th} stage of the experiments of the grad students;

- We set

$$\gamma_C = \mathbf{G}\,\mathrm{SD}_C \wedge \mathbf{G}\,\mathbf{Within}_{10}\,\mathrm{EMAIL},$$
$$\gamma_P = \mathbf{G}\,\mathrm{SD}_P \wedge \mathbf{G}\,\mathrm{WEB},$$
$$\gamma_E = \mathbf{G}\,\mathrm{SD}_E \wedge \mathbf{G}\,\mathrm{WEB};$$

- The endowment function of each agent is:

$$\mathrm{en}(C) = (2,1,20), \quad \mathrm{en}(P) = (4,1,10), \quad \mathrm{en}(E) = (2,2,10);$$

- The requirement function of each propositional variable is:

$$\mathrm{req}(\mathrm{SD}_i) = (1,0,0), \quad \mathrm{req}(\mathrm{EMAIL}) = (1,0,10),$$
$$\mathrm{req}(\mathrm{WEB}) = (1,0,10), \quad \mathrm{req}(\mathrm{GA}_k) = (4,2,0);$$

- Finally, we say that the grad students' deadline is given by the LTL specification,

$$\bigwedge_{k\in[100]} \mathbf{Within}_{3600}\,\mathrm{GA}_k.$$

  Informally, this specification can be thought of as the grad students completing their experiments within the hour.

With this in place, we can ask a question like 'does there exist some Nash equilibrium in which the grad students' deadline is met?'. Through observation, we can answer 'yes' to this question — the servers can undertake the experiments in the time between serving emails and web pages.[1]

[1]The eagle-eyed reader may note that a server may contribute CPU load to the core services of another server. This is perhaps not a realistic assumption in practice, and we see this happening because resources in IAGs are *fungible*. The example can be refined to reflect this by creating a new set of resources from the

We presented the previous example as an illustration of the sort of real-world systems that can be modelled within our framework. However, the above scenario could also feasibly be solved using standard model-checking algorithms. Where our framework really shines is when there is an adversarial component to the system. To this end, we present the following, slightly more fantastical, example of countries engaging in a space race:

Example 5. Suppose we have four countries, Arstotzka, Borduria, Chernarus and Dankmire. Each country has a selection of resources available to them each year, including US Dollars, metals and plastics. To keep their respective populaces happy, they each need to invest enough into domestic policy each year. However, Arstotzka and Borduria ideologically disagree with Chernarus and Dankmire over how a state should be governed; as such, they decide to compete in a space race in order to settle their differences. To complicate matters even further, the UN sets foreign aid targets for each of these countries. To model this, we create a IAG as follows:

- We set $\mathrm{Ag} = \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}$, representing the four counties;
- We set $R = \{\$, M, P\}$, representing US dollars, metals and plastics, respectively;
- We set $\mathrm{AP} = \{\mathrm{DP}_i, \mathrm{FA}_i \mid i \in \mathrm{Ag}\} \cup \{\mathrm{SR}_{j,k} \mid j \in \{\mathrm{AB}, \mathrm{CD}\}$, where DP_i represents the domestic policy of country i, FA_i represents country i contributing to the UN's foreign aid targets, and $\mathrm{SR}_{j,k}$ represents coalition j completing the k^{th} stage of their space programme;
- We set

$$\gamma_A = \mathbf{G}\, DP_A \wedge (\neg \mathrm{SR}_{\mathrm{CD}}\, \mathbf{U}\, \mathrm{SR}_{\mathrm{AB}}), \quad \gamma_B = \mathbf{G}\, DP_B \wedge (\neg \mathrm{SR}_{\mathrm{CD}}\, \mathbf{U}\, \mathrm{SR}_{\mathrm{AB}}),$$
$$\gamma_C = \mathbf{G}\, DP_C \wedge (\neg \mathrm{SR}_{\mathrm{AB}}\, \mathbf{U}\, \mathrm{SR}_{\mathrm{CD}}), \quad \gamma_D = \mathbf{G}\, DP_D \wedge (\neg \mathrm{SR}_{\mathrm{AB}}\, \mathbf{U}\, \mathrm{SR}_{\mathrm{CD}}),$$

Cartesian product of the players and the existing resources, and tweaking the endowment and requirement functions accordingly.

where

$$\mathrm{SR}_{\mathrm{AB}} = \bigwedge_{k\in[5]} \mathbf{F}\,\mathrm{SR}_{\mathrm{AB},k}, \quad \mathrm{SR}_{\mathrm{CD}} = \bigwedge_{k\in[5]} \mathbf{F}\,\mathrm{SR}_{\mathrm{CD},k};$$

- The endowment function of each agent is:

$$\mathrm{en}(A) = (2,1,1), \quad \mathrm{en}(B) = (3,0,1), \quad \mathrm{en}(C) = (2,1,0), \quad \mathrm{en}(D) = (3,0,1);$$

- The requirement function of each propositional variable is:

$$\mathrm{req}(\mathrm{DP}_i) = (1,0,0), \quad \mathrm{req}(\mathrm{FA}_i) = (1,0,0), \quad \mathrm{req}(\mathrm{SR}_{j,k}) = (2,1,1);$$

- Finally, we say that the UN's foreign aid targets are given by the LTL specification,

$$\mathbf{G} \bigwedge_{i\in\mathrm{Ag}} \mathrm{FA}_i.$$

With this in place, we can ask a question like 'does there exist some Nash equilibrium in which the UN's foreign aid targets are met?'. Through observation, we can answer 'no' to this question — if the foreign aid targets are met, then a party could deviate and complete their space programme quicker than the others.

In the preceding examples, we were able to determine whether there was some 'stable outcome' of the game solely through inspection. However, in general, this will not be possible — instead, we require some principled way of answering such questions. In the next sections, we will do precisely this: we will express the relevant questions as decision problems, and study their complexities.

Before proceeding, it is illustrative to compare our model with that of iterated Boolean games (IBGs) [101]. In an IBG, we have a set of agents, each of whom controls a set of propositional variables. At each turn, every player chooses some subset of their variables

to set as true; across all the agents, this induces a valuation. This process then repeats. Each player's goal is then given by an LTL formula, and players prefer runs where the resulting run of valuations models their formula over those that do not. IBGs can be seen as a special case of IAGs — suppose in an IBG, each player, i, controls a set of variables Φ_i with $|\Phi_i| = k_i$. Then we can create a IAG where each player has a unique resource r_i, with $\text{en}(i)_i = k_i$ and $\text{en}(i)_j = 0$ for all $j \neq i$. Then for each $p \in \Phi_i$, we set $\text{req}(p)_i = 1$ and $\text{req}(p)_j = 0$ for all $j \neq i$. As can be seen, this IAG exactly captures the behaviour of the original IBG, and is a polynomial time translation. As such, this immediately yields a number of lower bounds for the decision problems we are about to consider. Similarly, we will also reduce from a number of results of Gutierrez et al. [2019], proved in the context of LTL games — if one considers the subset of LTL games with fully-connected game graphs, then the problems remain **2EXPTIME**-complete, and offer a simple reduction. Additionally, some of the techniques we use in proving the upper bounds of our theorems were originally seen in the context of IBGs and LTL games.

4.3 Decision problems

Given a IAS S, the first question we can ask is whether a given strategy profile $\vec{\sigma}$ models some LTL formula φ:

> IAS-Model-Checking:
> *Given*: A IAS S, a strategy profile $\vec{\sigma}$, and an LTL formula φ.
> *Question*: Do we have $\rho(\vec{\sigma}) \vDash \varphi$?

Theorem 7. *With finite-memory strategies as input, IAS-Model-Checking lies in* **PSPACE**.

Proof. For membership, we begin by storing in memory the initial state of the Büchi automaton, along with the initial strategy state of each player. Then, we consider each player in turn, and use their choice function to determine what amount of resources they have

chosen to allocate to each propositional variable. We sum these vectors across the players, and then use the requirement function to determine which propositional variables are true under the total allocation, inducing a valuation v^0. Then for each player, we use the allocation along with their current strategy state to move to another strategy state, according to their state transition function. We also use the valuation to non-deterministically move to a new state in the Büchi automaton, respecting its transition function.

We repeat this process, deterministically traversing the strategy space, and non-deterministically traversing the automaton states. If we reach an accepting state in the Büchi automaton, we have a choice to non-deterministically store it, along with the current states of all the players strategies. We then continue to traverse the automaton and the strategy state space as before, until we reach the stored state again. This implies that there exists a path through the joint automaton-strategy space which visits an accepting state infinitely often. This in turns implies that $\rho(\vec{\sigma}) \vDash \varphi$. Note all of these manipulations can be done with polynomial space. In particular, the Büchi automaton can be constructed 'on-the-fly' and traversed with only polynomial space (this can be seen via a number of different constructions — one clear presentation is given in [203]). Thus, by Savitch's theorem [188], we may conclude that this procedure can be decided in **PSPACE**.

□

We now turn to decision problems relating to the solution concepts described earlier. The first is in the non-cooperative setting, and asks if a given profile is a Nash equilibrium:

NASH-MEMBERSHIP:
Given: A IAG G, and a strategy profile $\vec{\sigma}$.
Question: Is $\vec{\sigma}$ a Nash equilibrium of G?

Theorem 8. *With finite-memory strategies as input,* NASH-MEMBERSHIP *is* **PSPACE***-complete.*

Proof. To show membership in **PSPACE**, we begin by determining the winners and losers of G under $\vec{\sigma}$, *i.e.* checking whether $\rho(\vec{\sigma}) \vDash \gamma_i$ for each $i \in \mathrm{Ag}$. This can be done in **PSPACE**.

Then for each loser i, we need to check that if they had deviated, they still would not have been able to achieve their goal. We can do this in a similar fashion to the proof of Theorem 7: we fix the remaining players' strategies, and non-deterministically traverse both the game state space (consistently with the remaining players' strategies) and the Büchi automaton of the goal of player i. Similarly to before, we non-deterministically store an accepting state of the automaton, along with the current states of the remaining players' strategies. We then continue traversing the game graph until we hit upon the stored vector of states again. If this happens, then this shows that player i can deviate to achieve their goal, showing that the profile is not a Nash equilibrium. If instead we run this algorithm for each player, and we find that no player can deviate to achieve their goal, we may conclude that the profile is indeed a Nash equilibrium. Moreover, it is easy to see the above procedure can be done in **PSPACE**.

For the lower bound, we reduce from Nash-Membership for IBGs [101]. □

We can also look at a game with a cooperative lens, and ask what coalitions might form, as the players collectively try to achieve their objectives. Given a strategy profile, there may exist a coalition of players who would rather collectively deviate to achieve their goals. Thus, the following decision problem is natural:

Beneficial-Deviation:
Given: A IAG G, a strategy profile $\vec{\sigma}$, and a strategy vector $\vec{\sigma}'_C$.
Question: Is $\vec{\sigma}'_C$ a beneficial deviation from $\vec{\sigma}$?

Theorem 9. *With finite-memory strategies as input,* Beneficial-Deviation *is* **PSPACE***-complete.*

Proof. Similar to the proof of Theorem 8: first verify that $\rho(\vec{\sigma}) \not\models \bigwedge_{i \in C} \gamma_i$ with the IAS-Model-Checking algorithm; that is, none of the members of C achieve their goal under $\vec{\sigma}$. Then, as before, we fix the strategy vector $\vec{\sigma}'_C$, and non-deterministically traverse the (game states, player strategy states, Büchi automaton of $\bigvee_{i \in C} \neg\gamma_i$) state space. If we non-deterministically store a state vector and then revisit it, then we may conclude that $\mathrm{Ag} \setminus C$

has a strategy that ensures $\vec{\sigma}'_C$ is not a beneficial deviation for C. In this case we reject, otherwise we accept.

For the lower bound, we reduce from Beneficial-Deviation for LTL games [108]. □

Closely related is the following problem, which asks if a given strategy profile is a member of the core:

Core-Membership:
Given: A IAG G, and a finite-memory strategy profile $\vec{\sigma}$.
Question: Is $\vec{\sigma}$ a member of the core?

As discussed in Chapter 2, the results up until this point can be seen as the computationally 'easier' questions that one may ask about games — those that have a strategy profile provided as part of the input. As discussed in the previous section, we are primarily concerned with *finding* stable outcomes of IAGs (*i.e.* Nash equilibria, or members of the core) which model a given LTL specification — that is, we are looking to perform synthesis. There is an array of different decision problems relating to our setup, which we establish here. In the non-cooperative setting, the following problem is natural:

E-Nash:
Given: A IAG G, and an LTL formula φ.
Question: Does there exist some Nash equilibrium $\vec{\sigma}$ such that $\rho(\vec{\sigma}) \vDash \varphi$ holds?

The cooperative setting offers a richer set of decision problems, of which we will study the following:

Coalition-SAT:
Given: A IAS S, a coalition C, and an LTL formula φ.
Question: Does there exist a strategy vector $\vec{\sigma}_C$ such that $\rho(\vec{\sigma}_C, \vec{\sigma}_{\mathrm{Ag}\setminus C}) \vDash \varphi$, for all responses $\vec{\sigma}_{\mathrm{Ag}\setminus C}$?

and,

E-Core:

Given: A IAG G, and an LTL formula φ.

Question: Does there exist some member of the core $\vec{\sigma}$ such that $\rho(\vec{\sigma}) \vDash \varphi$ holds?

as well as the Core-Membership problem.

Before we analyse these problems, we need some more machinery. One natural idea is to convert everything to structures we already understand, namely concurrent game structures [20] and LTL games [95, 178, 184]. We do this as follows:

Lemma 8. *Let S be a IAS. Then there is a concurrent game structure, S^*, of size exponential in G, along with a bijective function which maps strategies on S to strategies on S^*. In IAGs, this mapping preserves LTL satisfaction, Nash equilibria, and members of the core.*

Proof. We begin by counting the number of possible allocations that a coalition C may have — if a player has $\mathrm{en}(i)_r$ of a resource r, then an allocation that uses all of that resource is a partition of $\mathrm{en}(i)_r$ into $|\mathrm{AP}|$ parts. Moreover, it is an *ordered* partition (*i.e.*, if we have 4 of a resource, then $1+3 \neq 3+1$). Thus, if we are looking to count allocations, it suffices to count the number of ordered ways a number n can be written as a sum of k numbers. We call this quantity $p_k(n)$. But if we want to count all possible allocations a coalition can do, then we also need to include all those allocations which do *not* allocate all resources (i.e. hold some back). This quantity is $p_k(n) + p_k(n-1) + p_k(n-2) + ...$, denoted by $t_k(n)$.

By thinking of n as items in a row, and k as delimiting indicators, we see that a closed form expression for $p_k(n)$ is precisely $\binom{n+k-1}{k-1}$. By the hockey stick formula, we have $t_k(n) = \binom{n+k}{k}$. So given a resource r, a coalition can allocate it in $\binom{\mathrm{en}(C)_r+|\mathrm{AP}|}{|\mathrm{AP}|}$ ways. Thus, a coalition C has,

$$\prod_{r \in R} \binom{\mathrm{en(C)}_r + |\mathrm{AP}|}{|\mathrm{AP}|}$$

possible allocations in total. Using a standard bound on the binomial coefficient, we have,

$$\binom{n}{k} < \left(\frac{n \cdot e}{k}\right)^k$$

Expanding, we get,

$$\begin{aligned}\prod_{r \in R} \binom{\text{en}(C)_r + |\text{AP}|}{|\text{AP}|} &< \prod_{r \in R} \left(\frac{e \cdot (\text{en}(C)_r + |\text{AP}|)}{|\text{AP}|}\right)^{|AP|} \\ &\leq \left(e \cdot \frac{\max_{r \in R} \text{en}(C)_r + |\text{AP}|}{|\text{AP}|}\right)^{|AP||R|}.\end{aligned}$$

As the set of atomic propositions is non-empty, we see the number of allocations is bounded by,

$$\left(e \cdot \left(\max_{r \in R} \text{en}(C)_r + |AP|\right)\right)^{|AP||R|}.$$

We can use this to obtain a bound in terms of the size of the game, $O(2^{p(|G|)})$, where p is some appropriately chosen polynomial. Using this information, we can then translate a IAS, denoted as $S = (\text{Ag}, \text{AP}, \{\gamma_i\}_{i \in \text{Ag}}, R, \text{en}, \text{req})$, to a concurrent game structure $M = (\text{Ag}', \text{St}, s^0, \{\text{Ac}_i\}_{i \in \text{Ag}}, \text{tr})$, defined in the following way:

- $\text{Ag}' = \text{Ag}$;
- $\text{St} = \{s_v \mid v \subseteq \text{AP}\} \cup \{s^0\}$;
- s^0 is a new state such that $s^0 \neq s_v$ for all $v \subseteq \text{AP}$;
- $\text{Ac}_i = \{\alpha_i \mid \alpha_i \in A_i\}$;
- $\text{tr}(s_{v_1}, \alpha) = s_{v_2}$ if and only if $\text{val}(\alpha) = v_2$; note the lack of dependence on s_{v_1}.

It follows that $|M| = 2^{|S|}$. Also, given the bijection between allocations of S and actions of M, we see that the mapping that takes a strategy of S, and returns a strategy on M, by simply replacing the output allocation with the corresponding action, must also be

a bijection. Additionally, if we have a IAG, G, we can form a corresponding LTL game G_{LTL} by mapping the IAS of G to a concurrent game structure, forming a labelling function $\lambda : \text{St} \to \mathscr{P}(\text{AP})$, by defining $\lambda(s_v) = v$, and keeping the players' goals the same. Whilst the concurrent game structure is exponentially larger than the original IAS, the LTL goals have not changed size.

Finally, it can be seen that the 'dynamics' of the original game are preserved in the translation, and in particular, that G has a strategy profile/Nash equilibrium/member of the core which models some specification, if and only if G_{LTL} has a strategy profile/Nash equilibrium/member of the core which models the same specification, preceded with **X**.

□

Given that the corresponding problems for IBGs and LTL games are **2EXPTIME**-complete in the size of the specifications, but polynomial in the size of the underlying structure, this yields the following results:

Theorem 10. *The following problems are all* **2EXPTIME***-complete:*

- *The E-Nash problem for IAGs;*
- *With finite-memory strategies as input, the Core-Membership problem for IAGs;*
- *The Coalition-SAT problem for IASs;*
- *The E-Core problem for IAGs.*

We note that whilst the lower bound for E-Nash comes from the corresponding problem for iterated Boolean games, the lower bounds for the remaining problems come from the corresponding problems for fully-connected LTL games.

One may ask if Lemma 8 is really necessary — part of the reason for the exponential blowup is that we take an input which has a logarithmic dependence on the number of resources an agent is endowed with (as this can be encoded in binary), and then performs

a translation based on this number itself (rather than its encoding). Can we map IAGs into IBGs without using this exponential transformation?

One possibility is to try and keep the endowment function in its original form as we translate it. What if we were to create a new IBG, where each agent controls variables corresponding to the bits of their allocations? That is, we could introduce variables $x_{i,p,r,j}$, each of which encodes the statement 'the allocation of player i to the propositional variable p with the resource r has a 1 in the j^{th} position of its binary expansion'. Then given a IAG,

$$G = (\mathrm{Ag}, \mathrm{AP}, R, \mathrm{en}, \mathrm{req}, \{\gamma_i\}_{i \in \mathrm{Ag}}),$$

we can then define a corresponding IBG,

$$G' = (\mathrm{Ag}', \{\mathrm{AP}'_i\}_{i \in \mathrm{Ag}}, \{\gamma'_i\}_{i \in \mathrm{Ag}})$$

as follows:

- $\mathrm{Ag}' = \mathrm{Ag}$;
- For each $i \in \mathrm{Ag}'$, we set $\mathrm{AP}'_i = \{x_{i,p,r,j} \mid p \in \mathrm{AP}, r \in R, j \leq \log_2 \mathrm{en}(i)_r\}$
- For each $i \in \mathrm{Ag}'$, we set $\gamma'_i = \gamma_i[p \to \mathrm{req}^*(p); p \in \mathrm{AP}]$

where the notation $\varphi[p \to \psi]$ denotes the formula obtained by replacing all instances of the propositional variable p in φ with the formula ψ, and the quantifier $p \in \mathrm{AP}$ states that this substitution should be done for each atomic proposition. Intuitively, the formula $\mathrm{req}^*(p)$ replaces each atomic proposition p with an encoding of all the allocations that make it true, according to the requirement function. Letting $\mathfrak{A}$ be the set

$$\mathfrak{A} = \bigtimes_{\substack{i \in \mathrm{Ag} \\ r \in R}} \mathscr{P}\left(\log_2 \mathrm{en}(i)_r\right),$$

and letting $\mathfrak{A}^* \subseteq \mathfrak{A}$ be the set

$$\mathfrak{A}^* = \{a \in \mathfrak{A} \mid \sum_{\substack{i \in \mathrm{Ag} \\ r \in R \\ j \in \alpha_{i,r}}} 2^j \geq \mathrm{req}(p)\},$$

then the formula $\mathrm{req}^*(p)$ is given by:

$$\bigvee_{\mathfrak{a} \in \mathfrak{A}^*} \bigwedge_{\substack{i \in \mathrm{Ag} \\ r \in R \\ j \in \alpha_{i,r}}} x_{i,p,r,j}$$

Note that in this translation, the exponential blowup has been moved from the structure of the game, to the new goals of the players. Since the exponential blowup in the goals has a linear number of atomic propositions, and the blowup only takes place in the propositional parts of the formula, not the temporal parts, the goals' corresponding Büchi automata will only exhibit polynomial blowup.

However, there is a fundamental flaw in the argument: in the resulting IBG, a player can 'over-allocate' — if a player has 1 unit of a resource available to them, with two atomic propositions, then in the IAG, they can either allocate that resource to the first atomic proposition, or to the second. However, in the associated IBG, there is nothing stopping that player setting the associated atomic proposition for each of these two allocations at the same time. As such, this may introduce new equilibria, or nullify existing equilibria.

One way we could potentially circumvent this is by extending IBGs with 'guards', either in the players' goals, or in the structure of the game, restricting the valid valuations for each player. If we were to include such an expression in the players' goals, then whilst a player would not want to take an action which could not be taken in the IAG, there is nothing actually stopping them from doing so — this introduces the possibility that a player might use the action in a punishing strategy, giving equilibria in the IBG which do

not exist in the IAG.

So for the idea of guards to work, we must embed them into the structure of the game: in addition to the usual setup of IBGs, we introduce a function $V : \mathrm{Ag} \to \mathscr{P}(\mathrm{AP})$ which says which valuations each player is allowed to play. The question then becomes, can we translate a IAG into a 'guarded IBG', without exponential blowup?

If we attempt to take this route, we run into another problem. As shown in Lemma 8, there are $\binom{\mathrm{en}(i)_r+|\mathrm{AP}|}{|AP|}$ possible allocations for player i (this is independent of how the endowment function is encoded). This is bounded from below by

$$\left(\frac{\mathrm{en}(i)_r + |\mathrm{AP}|}{|\mathrm{AP}|}\right)^{|\mathrm{AP}|} = \left(1 + \frac{\mathrm{en}(i)_r}{|\mathrm{AP}|}\right)^{|\mathrm{AP}|}.$$

Given that the exponential function can be defined as follows:

$$\exp(x) = \lim_{n\to\infty}\left(1 + \frac{x}{n}\right)^n,$$

we see that the number of allocations is bounded below by a quantity that grows like $\exp(\mathrm{en}(i)_r)$ as $|\mathrm{AP}|$ becomes large. Put more precisely, the number of allocations available to a player exceeds the growth of any fixed polynomial as $|AP|$ increases. Thus, for games whose size is dominated by its number of atomic propositions and its endowment functions, we see that a player will have allocations that grow superpolynomially with the size of the game. As such, we do not believe there to be an 'obvious' way of encoding these valid allocations with a restriction function without inducing some sort of superpolynomial blowup.

Thus, despite our best efforts, there does not appear to be an obvious polynomial translation from IAGs to IBGs. All such attempts end up with a 'lump in the carpet' — if you manage to remove the exponential dependency in one place, it just appears in another. We conjecture that no such translation exists, but a formal proof of this statement remains elusive.

Monotone LTL formulae

We can gather another set of insights by following a different route. Consider a Boolean formula, φ, along with its corresponding Büchi automaton, A_φ. We note that its transition relation, δ, will be of the form $\delta \subseteq Q \times \mathscr{P}(\mathrm{AP}) \times \mathscr{P}(Q)$. For every valuation $v \subseteq \mathrm{AP}$, we can form a linear set of equations and determine if the grand coalition can achieve it. Formally, for each $p \in v$ and each $i \in \mathrm{Ag}$, we introduce a variable $a_{i,p}$. Then for every $p \in v$, we can introduce the equation,

$$\mathrm{req}(p) \leq \sum_{i \in \mathrm{Ag}} a_{i,p}.$$

We also require that $a_{i,p} \geq 0$ for all such variables, along with,

$$\mathrm{en}(i) \geq \sum_{p \in v} a_{i,p}.$$

For a given $v \subseteq \mathrm{AP}$, we call this linear program ℓ_v; it is an integer linear program. We can then form a new Büchi automaton, which is identical to the original, but where we include an edge $(q, v, q') \in \delta$ if and only if ℓ_v is soluble. Note that we can do this using polynomial space. This new automaton constrains the original automaton by only including transitions that the grand coalition can effect. We can then use standard graph algorithms on this automaton to determine if there is a run that models the LTL formula.

Whilst this technique works for the grand coalition, it does not readily generalise to strict coalitions: if a coalition has enough resources to make a valuation true, it does not mean they can force that valuation — the counter-coalition could always make more variables true, potentially 'blocking off' the relevant transition in the Büchi automaton needed to trigger the acceptance condition. However, if we consider *monotone* LTL formulae, we can proceed as before. An LTL formula φ is monotone if for any π such that $\pi \vDash \varphi$, and for any π' where $\pi[k] \subseteq \pi'[k]$ for all $k \in \mathbb{N}$, then $\pi' \vDash \varphi$. In other words, if we have a monotone LTL formula, φ, and a coalition C can ensure that φ is satisfied without $\mathrm{Ag} \setminus C$

contributing anything, then no matter what allocations $\mathrm{Ag} \setminus C$ make on top of C's allocations, φ will still be satisfied. This is a reasonable assumption in many real-life systems, particularly when you have players or coalitions who are independent of one another.

We can encapsulate this as a decision problem,

> Monotone-Coalition-SAT:
> *Given*: A IAG G, a coalition C, and a monotone LTL formula φ.
> *Question*: Does there exist a strategy vector $\vec{\sigma}_C$ such that $\rho(\vec{\sigma}_C, \vec{\sigma}_{\mathrm{Ag}\setminus C}) \vDash \varphi$, for all responses $\vec{\sigma}_{\mathrm{Ag}\setminus C}$?

We can use the above technique to obtain the next result:

Theorem 11. *Monotone-Coalition-SAT is in* **PSPACE**.

In the context of propositional logic, monotone formulae have a nice characterisation: they are exactly those formulae that do not contain any negations. We do not know if there exists a similar characterisation for LTL formulae.

4.4 Questions about resources

Thus far we have taken a player-centric view of the games. But the other key component to our model is that of resources. Thus, it is natural to ask questions about the resources in the game, such as,

- Is a given resource necessary for a coalition to achieve some LTL formula?
- Can a coalition achieve an LTL formula with a bounded number of resources?
- If the players instead prefer runs which minimise resource expenditure, but which still model an LTL goal, how does that affect the strategic reasoning of the game?

One feature presents itself in the above list of questions: given a IAS S, a coalition C and a vector of resources $\vec{b} \in (\mathbb{N} \cup \{\infty\})^R$, it is often important to try and understand the game in which the coalition is constrained to use *at most* the amount of resources present in the vector. Formally, we can define a modified IAS, denoted $S[C, \vec{b}]$, consisting of two coalitional players C and $\mathrm{Ag} \setminus C$, each endowed with the resources of their respective players, but the resources available to coalition C are 'clamped' so that

$$\mathrm{en}(C)_r = \min\left(\sum_{i \in C} \mathrm{en}(i)_r, b_r\right).$$

We put no restrictions on the resources of $\mathrm{Ag} \setminus C$. Put differently, in $S[C, \vec{b}]$, coalition C can use all of a resource r available to its constituent agents, unless $b_r < \sum_{i \in C} \mathrm{en}(i)_r$, in which case they can only use b_r of the resource.

We begin by focussing on only one resource being restricted. One useful shorthand we adopt is $S[C, -r]$ for $S[C, (\infty, \ldots, 0, \ldots, \infty)]$, where the sole 0 in the resource vector lies in the r^{th} position. We then say that a resource $r \in R$ is *necessary* for $C \subseteq \mathrm{Ag}$ in the satisfaction of an LTL formula φ if 1. there is a strategy $\vec{\sigma}_C$ such that for all counter-strategies $\vec{\sigma}_{\mathrm{Ag}\setminus C}$, we have $\rho(\vec{\sigma}_C, \vec{\sigma}_{\mathrm{Ag}\setminus C}) \vDash \varphi$, and 2. in $S[C, -r]$, for all strategies $\vec{\sigma}_C$, there exists a counter-strategy $\vec{\sigma}_{\mathrm{Ag}\setminus C}$ such that $\rho(\vec{\sigma}_C, \vec{\sigma}_{\mathrm{Ag}\setminus C}) \nvDash \varphi$. Informally, a resource r is necessary for a coalition in satisfying an LTL formula if they cannot satisfy it when they are not allowed to use r, but they can satisfy it when the restriction is not in place. This definition suggests the following natural decision problem:

> NECESSARY-RESOURCE:
>
> *Given*: A IAS S, a coalition C, a resource $r \in R$, and a satisfiable LTL formula φ.
>
> *Question*: Is r necessary for C in the satisfaction of φ?

Theorem 12. *NECESSARY-RESOURCE is* **2EXPTIME***-complete.*

Proof. We first consider S and use Coalition-SAT to see if C can achieve φ when their resources are not restricted (other than by their endowment functions). If C can achieve φ, then we again use Coalition-SAT with $S[C, -r]$ to see if C can achieve φ in this new IAS. If they cannot, then for all strategies of C, there exists a counter-strategy of $\text{Ag} \setminus C$ that prevents C from achieving φ. By definition, this means that r is a necessary resource for C in satisfying φ. The lower bound follows from Coalition-SAT. □

We may now generalise to restricting multiple resources at once, along with these restrictions being arbitrary (as opposed to the agents either being allowed to use the resource, or not). Formally, we say that C can satisfy φ using at most $\vec{b}$ resources if in $S[C, \vec{b}]$, there exists some strategy $\vec{\sigma}_C$ such that for all counter-strategies $\vec{\sigma}_{\text{Ag}\setminus C}$, we have $\rho(\vec{\sigma}_C, \vec{\sigma}_{\text{Ag}\setminus C}) \vDash \varphi$. The following decision problem asks whether a coalition can achieve an LTL formula whilst constrained by a given bound:

> Resource-Bound-SAT:
> *Given*: A IAS S, a coalition C, a resource bound $\vec{b} \in (\mathbb{N} \cup \{\infty\})^R$, and a satisfiable LTL formula φ.
> *Question*: Can C satisfy φ using at most $\vec{b}$ resources?

Theorem 13. *Resource-Bound-SAT is* **2EXPTIME***-complete.*

Proof. In a similar way to Theorem 12, we use Coalition-SAT to see if C can achieve φ in $S[C, \vec{b}]$. If they can, then by definition, C can satisfy φ using at most $\vec{b}$ resources. For the lower bound, we reduce from Coalition-SAT. □

Note that the two preceding decision problems are stated solely in terms of IASs: they are independent of agent preferences. We now reintroduce agents preferences, modified from our original IAGs to take resource expenditure into account and study the resulting game.

In IAGs, agents have dichotomous LTL preferences — under a given run, their goal is satisfied, or it is not. However, this is a rather coarse-grained view of events: it is not unreasonable to imagine that an agent might want to *minimise* the total amount of resources it is using when trying to achieve its goal. Thus, we look to incorporate this added dimension to our agent preferences.

If agents only had one resource available to them, then defining a preference relation where an agent wants to minimise the number of resources they are using, whilst ultimately trying to satisfy a given LTL formula, is easy — players are indifferent between runs which do not satisfy their LTL goal, prefer runs which satisfy their LTL goal over ones which do not, and among runs which do satisfy their LTL goal, prefer those that minimise the mean-cost of their resource usage (defined over allocations).

However, when an agent has multiple resources available to them, the route to take is less obvious. Options include restricting attention to a particular resource; preferring runs where resource usage is under a threshold; assigning weights to resources, and taking the dot product; or ordering resources lexicographically. All of these approaches have benefits and limitations, and rather than commit to a particular route of attack, we abstract away and offer another alternative.

For each agent, we assume the existence of a cost function, $c_i : \mathbb{N}^R \to \mathbb{Q}$. We can extend cost functions to allocations by defining,

$$c_i(\alpha_i) = \sum_{p \in \mathrm{AP}} c_i(\alpha_i(p)),$$

for an allocation α_i. Then given a run $\vec{\alpha}$, we can define its cost, $c_i(\vec{\alpha})$, to be $\mathrm{mc}(c_i(\vec{\alpha}[t]))$, using the *mean-cost*, as defined in Chapter 2.

With this in place, we are ready to introduce a new type of game, namely *resource-*

minimisation iterated allocation games (RMIAGs). An RMIAG, G, is a tuple,

$$G = (S, \{\gamma_i\}_{i \in \mathrm{Ag}}, \{c_i\}_{i \in \mathrm{Ag}}),$$

where

- S is a IAS;
- For each player i, γ_i is an LTL goal;
- For each player i, $c_i : \mathbb{N}^R \to \mathbb{Q}$ is a cost function.

RMIAGs follow the standard mechanism of game playing as detailed in Section 4.2. However, for each player i, the preference relation $\succ_i$ is defined as follows,

- If $\vec{\alpha}, \vec{\alpha}'$ are two runs such that $\mathrm{val}(\vec{\alpha}) \not\models \gamma_i$ and $\mathrm{val}(\vec{\alpha}') \not\models \gamma_i$, then we have $\vec{\alpha} \sim_i \vec{\alpha}'$;
- If $\vec{\alpha}, \vec{\alpha}'$ are two runs such that $\mathrm{val}(\vec{\alpha}) \models \gamma_i$ but $\mathrm{val}(\vec{\alpha}') \not\models \gamma_i$, then $\vec{\alpha} \succ_i \vec{\alpha}'$;
- If $\vec{\alpha}, \vec{\alpha}'$ are two runs such that $\mathrm{val}(\vec{\alpha}) \models \gamma_i$ and $\mathrm{val}(\vec{\alpha}') \models \gamma_i$, and $c_i(\vec{\alpha}) < c_i(\vec{\alpha}')$, then $\vec{\alpha} \succ_i \vec{\alpha}'$.

Note that common ideas of resource minimisation, such as all of those given above, can be encoded with this model:

- To restrict attention to a particular resource, r_i, we can define $c_i(r_1, \ldots r_k) = r_i$;
- To prefer runs where resource usage is under a given threshold, $\vec{b} \in \mathbb{N}^R$, we can define $c_i(\vec{r}) = 1$ if there exists some i such that $r_i > b_i$, and have $c_i(\vec{r}) = 0$ otherwise;
- To weight each resource, r_i, we introduce a vector of weights, $\vec{w}$ and then define $c_i(\vec{r}) = \vec{w} \cdot \vec{r}$;

- Given a preference/ordering over the resources $r_{i_1} > ... > r_{i_k}$, we can define a lexicographic cost over these resources by letting $B = \max_{r \in R} \text{en}(i)_r + 1$, and then defining,

$$c_i(\vec{r}) = \sum_{j=0}^{k} r_{i_j} B^{k-j}.$$

With this new preference relation in place, we can ask questions about it, such as whether a given strategy profile is a member of a given solution concept, or whether a game has *any* members of a given solution concept. Given the richness of this new preference relation, we leverage a result from existing work — as in Gutierrez et al. [2020], we ask if a given RMIAG has a *finite-memory, strict ε-Nash equilibrium*. We defined finite-memory strategies in Section 4.2; we define strict ε-Nash equilibria here. Given $\varepsilon > 0$, we say that a strategy profile $\vec{\sigma}$ is a strict ε-Nash equilibrium if for all agents i, and for all strategies σ_i', we have that when $\rho(\vec{\sigma}_{-i}, \sigma_i') \vDash \gamma_i$, then $\rho(\vec{\sigma}) \vDash \gamma_i$ and $c_i(\rho(\vec{\sigma})) < c_i(\rho(\vec{\sigma}_{-i}, \sigma_i')) + \varepsilon$. These extra restrictions (strictness; the added ε) on top of the 'vanilla' Nash equilibrium come from the fact that in mean-payoff parity games [65], infinite memory may be required to act optimally, whilst acting ε-optimally can be done with finite memory [41].

We are now ready to introduce the key decision problem:

> E-FSNE$^{\varepsilon}$:
>
> *Given*: An RMIAG G, and an LTL formula φ.
>
> *Question*: Does there exist some finite-memory, strict ε-Nash equilibrium, $\vec{\sigma}$, such that $\rho(\vec{\sigma}) \vDash \varphi$?

Theorem 14. *E-FSNE$^{\varepsilon}$ is* **2EXPTIME***-complete.*

Proof. For the upper bound, use Lemma 8 to yield a concurrent game structure of exponential size, but with LTL goals still the same size. We then note that minimising the mean-cost of allocations is equivalent to maximising the mean-payoff of their negations.

Then, we follow through the proof steps of [112], making sure to account for the RMIAG preference relation, rather than the lexicographic preference relation of the original proof. This yields the result. For the lower bound, we reduce from the same problem. □

Chapter 5

Voting games

5.1 Introduction

In this chapter, we will consider another method for modelling agents with non-exclusive control of a system: voting. Voting is a natural mechanism for resolving disagreements over what action should be taken in a given situation, and is easily incorporated into a game setting. Our games are built on top of transition systems [128], which players traverse by voting on what action to take next. We use a form of approval voting, where players are each endowed with a number of voting credits which they can allocate across the possible actions available at each state of the transition system. Players are also each equipped with an Emerson-Lei condition for a goal, which they aim to satisfy with respect to runs of the transition system.

These games offer a number of novelties to the rational verification paradigm: the voting mechanism induces non-determinism in the games, and the Emerson-Lei conditions a sub-**2EXPTIME** complexity. As seen in the previous chapter, **2EXPTIME** results are a blessing and a curse: one can perform a lot of additional pre- or post-processing and still remain in **2EXPTIME**, which allows us to often take the most obvious (naive?) approach. However, this is a double-edged sword: the high complexity often 'hides' the

interesting features of the pre- and post-processing. This motivates our decision to work with Emerson-Lei conditions for our goals, rather than LTL. We will show that in our setting, Emerson-Lei synthesis lies within the polynomial hierarchy, surprisingly beating the **PSPACE**-completeness bound for classical Muller games. We will then leverage this result in our analysis of multiplayer games, deriving sub-**2EXPTIME** complexity bounds for the usual rational verification decisions problems for both the Nash equilibrium and the core

This chapter is structured as follows: we begin in Section 5.2, with a simpler game model, to whet the reader's appetite and to motivate the remainder of the chapter. In Section 5.3, we introduce our computational model, namely that of *transition systems*, a relatively simple, widely used formalism for modelling always-live systems. We then introduce our main game model, *iterated voting games* in Section 5.4, providing the necessary setup along with an example, before taking a detour through two-player, zero-sum iterated voting games in Section 5.5 — we will have to appeal to this latter type of games to help us understand the former. Resuming the discussion of the multiplayer case in Section 5.6, we establish the complexities of a number of decision problems in both the non-cooperative and the cooperative settings. Finally, we finish with a modification of the basic model, one which allows path quantifiers within the players' goals, in Section 5.7

5.2 One-shot games

As a warm-up, we start with one-shot voting games. A Boolean voting game (BVG) is a tuple,

$$G = \left(N, \mathrm{AP}, \{c_i\}_{i \in N} \{\gamma_i\}_{i \in N}\right),$$

where,

- N is a finite, non-empty set of *agents*;

- AP is a finite, non-empty set of *atomic propositions*;
- For each agent $i \in N$, $c_i \in \mathbb{N}$ is the number of *voting credits* available to them;
- For each agent $i \in N$, γ_i is a propositional formula over AP, called the *goal* of i.

The size of a game G, denoted $|G|$, is given by the expression,

$$|N| + |\text{AP}| + \sum_{i \in N} \log_2 c_i + \sum_{i \in N} |\gamma_i|$$

Informally, a *vote* for a player is an assignment of their voting credits across the atomic propositions. Formally, a vote for a player i is a function, $v_i : \text{AP} \to \mathbb{Z}$, subject to the constraint,

$$\sum_{p \in \text{AP}} |v_i(p)| \leq c_i$$

This condition simply ensures that one cannot use more voting credits than they possess. A vote vector v comprises a vote for each player; if it is apparent from the context, we may just refer to a vote vector as a vote. The game is then simple: each player casts a vote (*i.e.* they choose how many voting credits they wish to assign for or against each atomic proposition), and then the votes are tallied. If there are more votes for than against an atomic proposition, then it is interpreted to be true. Otherwise, we interpret it to be false. We denote the set of all votes available to a player i by V_i.

Formally, for a given atomic proposition $p \in \text{AP}$, and a vote vector v, we say that v *models* p and write $v \vDash p$ if we have,

$$\sum_{i \in N} v_i(p) > 0.$$

Over all the atomic propositions, this induces a valuation, $\text{val}(v)$, in the obvious way:

$$\text{val}(v) = \{p \in \text{AP} \mid v \vDash p\}.$$

We can then interpret the players' goals relative to this valuation. Specifically, we say that a player i is a *winner* under v if $\text{val}(v) \vDash \gamma_i$, and call them a *loser* otherwise. This in turn helps us define player preferences over votes: each player prefers to be a winner over a loser, and is indifferent between votes which make them winners, and indifferent between votes which make them losers.

Let us consider an example which should make this all clearer, and will motivate the next section in more detail.

Example 6. Suppose we have a (small) legislature, with 5 members. Moreover, suppose there are a number of pertinent issues they wish to vote on, such as healthcare, education, defence, policing and fishing rights, but each member has only three voting credits available to them, and must decide which issues matter the most to them. Identify N with the set $\{1, \dots, 5\}$, and let $\text{AP} = \{\text{HC}, \text{ED}, \text{DF}, \text{PL}, \text{FR}\}$ be a set of atomic propositions corresponding with the aforementioned issues. We set $c_i = 3$ for each $i \in N$, and we set the goals as follows:

$$\gamma_1 = \text{HC} \wedge \text{ED} \wedge \text{DF}$$

$$2\gamma_2 = \text{HC} \wedge \text{ED} \wedge \text{PL}$$

$$\gamma_3 = \neg\text{HC} \wedge \text{DF} \wedge \text{PL}$$

$$\gamma_4 = \text{DF} \wedge \text{PL} \wedge \text{FR}$$

$$\gamma_5 = \text{ED} \wedge \text{FR}$$

Now suppose we are interested in ensuring that the specification $\text{HC} \wedge \text{ED}$ is satisfied. Does there exist some Nash equilibrium which models this specification, and if so, do all such Nash equilibria do so? We can answer the first question in the affirmative: consider

the outcome where the players vote as follows (where we write a vote v_i as a vector $\mathbb{Z}^{\mathrm{AP}}$):

$$v_1 = (1,1,1,0,0)$$
$$v_2 = (1,1,0,1,0)$$
$$v_3 = (-1,1,1,0,0)$$
$$v_4 = (0,0,1,1,1)$$
$$v_5 = (1,1,0,01)$$

Under these votes, we see that each player gets their goal achieved, except for player 4. Moreover, even if player 4 were to change their vote to something else, they still would not get their goal achieved. Thus, this is a Nash equilibrium which models the specification. Additionally, by the same reasoning, it is also a member of the core which models the specification.

However, if player 5 were to change their vote to $v_5 = (-1,0,0,1,1)$, we see that under this new strategy profile, players $3,4$ and 5 get their goals achieved, whilst players 1 and 2 do not. As before, they have no way of changing their votes to force their goals. So we also have a Nash equilibrium which does *not* model the specification. We also note that it is a member of the core which does not model the specification: we have accounted for individual deviations, so we just need to consider any possible joint deviation by player 1 and 2. This is easy to see: players 1 and 2 have six votes between them, whilst player 3, 4 and 5 have nine votes. So regardless of how $\{1,2\}$ cast their votes, the counter-coalition can make sure they do not get their goal achieved by putting -9 votes into healthcare. Thus, we answer our latter question in the negative: it is *not* the case that all equilibria model the specification.

As in our previous settings, this example was simple enough that we could reason about it by hand. However, we are interested in understanding arbitrary games, which

may be very large, and impossible to analyse by hand. So again, we formalise the questions we wish to ask about our games as decision problems. Just as a 'taster', we consider one decision problem in this setting, namely the E-NASH problem:

Theorem 15. *E-NASH is Σ_2^P-complete.*

Proof. We guess a vote for each player in **NP**, see which players win under this vote, and then show that no player has any incentive to deviate, using a **co-NP** oracle to guess a deviating player and vote. A vote is of at most size $|\text{AP}| \cdot \log_2 \max_{i \in N} c_i$, so can be guessed in polynomial time.

Hardness follows by a reduction to Boolean games [47]. □

This result is unsurprising — it matches the original result in the context of Boolean games [47], and uses an identical proof technique. However, in the following section, we will consider iterated games, a natural formalism for modelling concurrent systems; here, we will have to develop new proof techniques and will derive new results.

5.3 Transition systems

We use transition system as our underlying computational model, on top of which we can introduce players with voting credits and goals, giving us a game. A transition system, TS, is a tuple,

$$\text{TS} = (\text{St}, \text{Ac}, \text{tr}),$$

where,

- St is a finite, non-empty set of states;
- Ac is a finite, non-empty set of actions;
- $\text{tr} \subseteq \text{St} \times \text{Ac} \times \text{St}$ is a labelled transition relation which takes a state and an action, and produces a new state;

- tr is left-total in its first element; that is, for every $s \in \mathrm{St}$, there exists some $\mathrm{ac} \in \mathrm{Ac}$ and $t \in \mathrm{St}$ such that $(s, \mathrm{ac}, t) \in \mathrm{tr}$;
- tr is functional in its third element; that is, for every $s \in \mathrm{St}$ and $\mathrm{ac} \in \mathrm{Ac}$, there is at most one s' such that $(s, \mathrm{ac}, s') \in \mathrm{tr}$.

The requirement that the transition relation in a transition system is left-total and functional gives us two nice properties. First, there is always a possible transition; the system cannot get 'stuck' in a certain state. Second, paths within the system are entirely deterministic, and do not branch. We will introduce non-determinism to our games, but through the players' actions rather than through the underlying system.

Given the restrictions on our transition relation, we can also view it as a partial function; that is, if $(s, \mathrm{ac}, t) \in \mathrm{tr}$, we may write $\mathrm{tr}(s, \mathrm{ac}) = t$. We can go further: (with an abuse of notation) we can regard Ac as a function, which gives the actions which are available in a given state. Formally, given a state $s \in S$, we define $\mathrm{Ac}(s)$ in the obvious way:

$$\mathrm{Ac}(s) = \{\mathrm{ac} \mid \exists \mathrm{ac} \in \mathrm{Ac}, \exists t \in \mathrm{St} \text{ s.t. } (s, \mathrm{ac}, t) \in \mathrm{tr}\}.$$

A transition system should then be interpreted as follows: from a start state s^0, given an action ac^0 from $\mathrm{Ac}(s^0)$, the transition system moves to the state $s^1 = \mathrm{tr}(s^0, \mathrm{ac}^0)$ and this then repeats ad infinitum. Thus, if s^n is the n^{th} state visited, then given an action $\mathrm{ac}^n \in \mathrm{Ac}(s^n)$, the $(n+1)^{\text{th}}$ state visited is $s^{n+1} = \mathrm{tr}(s^n, \mathrm{ac}^n)$. This in turn defines a path $\pi = s^0 s^1 s^2 \ldots$. Conversely, we say a path $\pi = s^0 s^1 s^2 \ldots$ is *consistent* with tr if there exists a run of actions $\mathrm{ac} = \mathrm{ac}^0 \mathrm{ac}^1 \mathrm{ac}^2 \ldots$ such that $s^{n+1} = \mathrm{tr}(s^n, \mathrm{ac}^n)$; we say that a path π is consistent with a transition system TS if π is consistent with tr(TS).

We could proceed as in Chapter 4, and use LTL formulae to describe qualitative behaviours of the transition system, but this would invariably lead to all the decision problems that we usually consider having the 'expected' complexities: **PSPACE**-completeness for model-checking-like problems, and **2EXPTIME**-completeness for synthesis-like prob-

lems. Instead, we will use Emerson-Lei conditions to reason about transition systems: this should induce lower complexities in our usual set of rational verification decision problems, and will require a different, more nuanced set of proof techniques (rather than just 'blowing everything up', as we do in the LTL/**2EXPTIME** setting).

It is clear that runs over transition systems can be used to interpret Emerson-Lei conditions (using the Emerson-Lei semantics from Chapter 2) — we will use this to define player goals in the next section. Before we do that and introduce players to the game, we study the transition system in their own right with respect to Emerson-Lei conditions. First we supply two definitions: we say that an Emerson-Lei condition, α, is *satisfiable* over TS from s if there exists some path π consistent with TS and starting at s such that $\pi \vDash \alpha$; we say that it is *valid* over TS from s if for all paths π consistent with TS and starting at s we have that $\pi \vDash \alpha$. It should be clear that α is valid over TS from s if and only if $\neg\alpha$ is not satisfiable over TS from s. We can then ask whether a given Emerson-Lei condition is satisfiable or valid over a transition system:

Emerson-Lei-SAT:
Given: A transition system TS, a state $s \in$ St(TS), and an Emerson-Lei condition α.
Question: Is α satisfiable over TS from s?

Emerson-Lei-Tautology:
Given: A transition system TS, a state $s \in$ St(TS), and an Emerson-Lei condition α.
Question: Is α valid over TS from s?

Theorem 16. *Emerson-Lei-SAT is* **NP***-complete, and Emerson-Lei-Tautology is* **co**-**NP***-complete.*

Proof. We start with Emerson-Lei-SAT: for membership, we guess a satisfying assignment for α, which is a set $X \subseteq$ St, and verify that any run π with permset$(\pi) = X$ is such that

$\pi \vDash \alpha$. If this is the case, then we check if any state in X is reachable from s: this can be done in polynomial time using depth-first search. If so, then we check that every element of X is contained within the same strongly connected component — this can be done using Tarjan's algorithm [200], which can be performed in linear time. If this is the case, then we accept; otherwise, if any of the aforementioned checks fail, then we reject.

For hardness, we reduce from SAT [76, 145]: given a propositional formula φ over a set of atomic proposition AP, we form a transition system TS = (St, *Ac*, tr) as follows:

- St = AP;
- Ac = AP;
- $\mathrm{tr}(*, p) = p$.

So our transition system looks like a fully-connected graph with each node labelled by a different atomic proposition, and the transition system moves to a given atomic proposition from any state if the corresponding atomic proposition action is taken. Additionally, we set s to any state arbitrarily, and we let α be the Emerson-Lei condition produced by transforming every instance of $p \in \mathrm{AP}$ in φ with $\mathrm{Inf}(p)$, whilst keeping the logical connectives the same. So, for instance, if $\varphi = p \vee q$, then α will be $\mathrm{Inf}(p) \vee \mathrm{Inf}(q)$.

We claim that φ is satisfiable if and only if α is satisfiable over TS from s. If φ is satisfiable, then it is easy to see that α is satisfiable over TS from s — take the satisfying valuation $v = \{p_1, \dots, p_k\}$ for φ, and consider the run $\rho = s\overline{p_1 \dots p_k}$. By construction, it should be clear that ρ models α, is consistent with TS and starts at s. Thus, we have that α is satisfiable over TS from s. For the other direction, suppose that α is satisfiable over TS from s. Then there exists some path π, starting at s, consistent with TS, such that $\pi \vDash \alpha$. Let $v = \mathrm{permset}(\pi)$; again, it should be clear by construction that $v \vDash \varphi$, *i.e.* φ is satisfiable.

For Emerson-Lei-Tautology, recall that α is valid over TS if and only if $\neg\alpha$ is not satisfiable over TS: this gives us that Emerson-Lei-Tautology is co-NP-complete. □

5.4 Iterated voting games

With transition systems defined, we are ready to build our games on top of them. An iterated voting game, is a tuple,

$$G = (\mathrm{TS}, s^0, \mathrm{Ag}, \{c_i\}_{i \in \mathrm{Ag}}, \{\gamma_i\}_{i \in \mathrm{Ag}})$$

where,

- TS is a transition system;
- $s^0 \in \mathrm{St}(\mathrm{TS})$ is the start state of the game;
- Ag is a finite, non-empty set of players;
- For each player i, we have that $c_i \in \mathbb{N}$ is a positive, integral number of *voting credits*;
- For each player i, we have that γ_i is an Emerson-Lei goal over St(TS).

The game proceeds as follows: the game begins in the start state s^0. Every player casts their vote, which is now distributed over the actions of the transition systems, rather than atomic propositions like in Section 5.2. Specifically, a vote for a player i, is a function $v_i : \mathrm{Ac} \to \mathbb{Z}$ subject to the constraint,

$$\sum_{a \in \mathrm{Ac}} |v_i(a)| \leq c_i.$$

We use the notation V_i to refer to the set of votes available to player i, and V to refer to the Cartesian product $V_1 \times \cdots \times V_{|\mathrm{Ag}|}$.

Given a set X and a function $f : X \to \mathbb{Z}$, we define the arg max of f to be the subset of X whose elements attain the maximum value of f. Formally, we have,

$$\arg\max(f) = \{x \in X \mid f(x') \leq f(x) \text{ for all } x' \in X\}$$

The votes are then tallied; intuitively, the most natural thing to do would be to take the action which receives the most votes. However, as a vote can result in a tie, this results in some non-determinism. Moreover, we need to account for the fact that a player may have voted for an action which is not available in a given state (a 'spoiled ballot'). So given a vote vector $v : \mathrm{Ag} \to \mathrm{Ac} \to \mathbb{Z}$ and a state s, the winning actions of v in s are given by,

$$\mathscr{W}(v, s) = \arg\max_{\mathrm{ac} \in \mathrm{Ac}(s)} \sum_{i \in \mathrm{Ag}} v_i.$$

It should be apparent that this set is always non-empty. An action, a^0, is chosen from this set, and the game moves to the state $s^1 = \mathrm{tr}(s^0, a^0)$. This process then repeats indefinitely. Now, due to the non-determinism, it should be clear that this process does not result in a linear run of states or actions, but rather a tree. So how do we interpret the players' goals, and more generally, Emerson-Lei conditions, over these trees?

As we defined them in Chapter 2, Emerson-Lei conditions have path-based semantics; in Chapter 3, we used them to reason effectively about deterministic systems; in the previous section, we studied them relative to paths over transition systems. We can leverage our previous definitions to define them relative to trees, but first we need to make a choice: should Emerson-Lei condition satisfaction on trees be *universal* or *existential*? Informally, we can say that a tree universally models an Emerson-Lei condition if every branch of the tree models the condition; conversely, we say that a tree existentially models an Emerson-Lei condition if there exists some branch of the tree which models the condition.

Formally, for an Emerson-Lei condition α over a set of states St, and a St-labelled tree τ, we say that τ *universally* models α, denoted $\tau \models_\forall \alpha$ if for every branch π of τ, we have $\pi \models \alpha$. Conversely, we say that τ *existentially* models α, and write $\tau \models_\exists \alpha$ if for some branch π of τ, we have $\pi \models \alpha$.

Universal satisfaction is useful when reasoning about properties like safety and liveness — when you want to make sure that no matter what behaviour the non-determinism

in the system induces, certain guarantees will hold no matter what. Conversely, existential satisfaction is useful when you want to know if some outcome is possible in a system. For our purposes, we will work primarily with universal satisfaction; however, existential satisfaction will implicitly play a key part in our arguments.

So, given a tree τ, we will either have that τ universally models an Emerson-Lei condition α, or that it does not. We can use this to give our players preferences over trees: for a tree τ, if $\tau \vDash_\forall \gamma_i$, then we say that player i is a *winner* under τ; otherwise, they are a *loser*. Players prefer trees in which they are winners over trees in which they are losers, but are indifferent between trees which make them winners, and indifferent between trees which make them losers.

Strategies in this setting are 'meta-actions' of the game. In full generality, they are functions $\sigma_i : V^+ \to V_i$; that is functions that take the history of the game so far (that is, the voting record of each player), and produce a vote that the player should cast. A strategy profile $\vec{\sigma}$ induces a tree in the game, $\tau(\vec{\sigma})$, in the obvious way: the root of the tree is s^0, and the players each produce a vote $v_i^0 = \sigma_i(\varepsilon)$. Together this forms a vote vector v^0, and produces a set of winning actions $\mathscr{W}(v^0, s)$. If $|\mathscr{W}(v^0, s)| = m$, then for each action in this set, $\mathscr{W}(v^0, s)$, which we identify with $\{\mathrm{ac}^{0,j}\}_{j\in[m]}$, then the children of s^0 are $\{\mathrm{tr}(s^0, \mathrm{ac}^{0,j})\}_{j\in[m]}$. For each of these children, we recursively repeat this procedure, yielding a tree $\tau(\vec{\sigma})$.

Let us now consider an example, to help illustrate the formalism as presented thus far:

Example 7. Suppose we have a transition system with six states, $\mathrm{St} = \{s_j\}_{j\in[0..5]}$, two actions, $\mathrm{Ac} = \{a, b\}$ and a transition relation tr with the following transitions:

$$(s_0, a, s_1), \quad (s_0, b, s_2), \quad (s_1, a, s_1), \quad (s_2, a, s_3), \quad (s_3, a, s_4),$$
$$(s_3, b, s_4), \quad (s_4, a, s_2), \quad (s_4, b, s_5), \quad (s_5, a, s_2), \quad (s_5, b, s_0)$$

On top of this transition system, we designate s_0 to be the start state, and introduce three players, $\mathrm{Ag} = \{1, 2, 3\}$. We then endow players 1 and 2 with 1 voting credit each, and player 3 with 2 voting credits. Finally, we set player 1's goal to be $\gamma_1 = \mathrm{Inf}(s_1)$, player 2's to be $\gamma_2 = \mathrm{Inf}(s_3)$ and player 3's to be $\gamma_3 = \mathrm{Fin}(s_2) \vee (\mathrm{Inf}(s_4) \wedge \mathrm{Fin}(s_5))$. Thus, the game is as depicted in Figure 5.1.

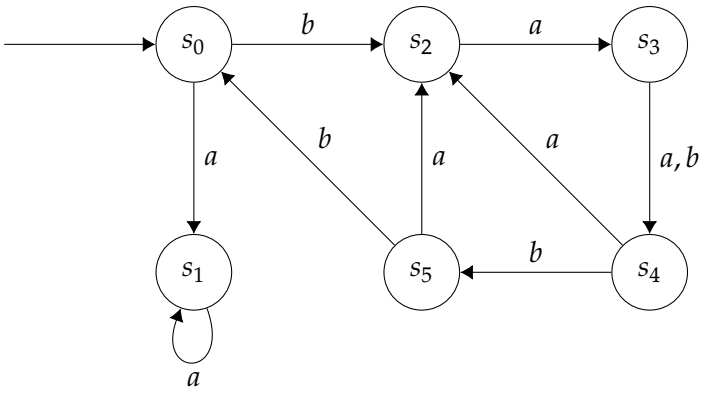

Figure 5.1: An example of an iterated voting game with three players.

Consider any strategy profile where player 1 casts one vote for a in s_0 and player 3 casts two votes for a in s_0. Then no matter how player 2 votes in s_0, the game will move to s_1 and remain there forever. This will induce a tree τ with only one path, namely $\pi = s_0 s_1 s_1 \ldots$. Thus, we have $\tau \vDash_\forall \gamma_1$ and $\tau \vDash_\forall \gamma_3$, but also $\tau \nvDash_\forall \gamma_2$. This is a Nash equilibrium: no matter how player 2 changes their vote, they cannot stop play entering s_1; this also makes it a member of the core.

Now (with our system designer hats on) suppose that s_1 is some sort of 'undesirable' state. Instead, we may want to know if there exists some equilibrium $\vec{\sigma}$ which models $\mathrm{Inf}(s_2)$ (a sort of liveness property). And indeed, we can answer this in the affirmative: let the players collectively vote so that they follow the path $s_0\overline{s_2 s_3 s_4 s_2}$. Then under this strategy, we have players' 2 and 3 goals satisfied, player 1's goal unsatisfied, and the specification $\mathrm{Inf}(s_2)$ modelled. Moreover, given the collective number of votes of players 2 and 3, player 1 cannot change their vote to achieve their goal, making this both a Nash

equilibrium and a member of the core.

As before, our analysis in the above example was done by hand; we would like instead to have algorithms with complexity guarantees. However, before we explore the usual rational decision problems in this setting, we will need to go via a simpler model, that of two-player, zero-sum iterated voting games.

5.5 Two-player, zero-sum iterated voting games

To be able to characterise multiplayer games, we will first consider a restricted form of them — two-player, zero-sum games. Ideally, we would like to be able to take our described formalism, set the number of players to be 2, and set $\gamma_2 = \neg\gamma_1$. However, as satisfaction of player goals is defined relative to trees, one can easily construct a two player game and a tree over that game, where one player's goal is the negation of the other's, and the tree models neither of the player's goals. We encapsulate this in the following example.

Example 8. Let $\mathrm{TS} = (\mathrm{St}, \mathrm{Ac}, \mathrm{tr})$ be a transition system, with three states $\mathrm{St} = \{s, t, u\}$, two actions $\mathrm{Ac} = \{a, b\}$, and a transition relation tr, containing the following transitions:

$$(s, a, t), \quad (s, b, u), \quad (t, a, s),$$
$$(t, b, s), \quad (u, a, s), \quad (u, b, s).$$

We build a game on top of this, where we let s be the start state, and introduce two players, $\mathrm{Ag} = \{1, 2\}$, each of whom has just one voting credit each ($c_1 = 1, c_2 = 1$). Finally, we set player one's goal to be $\gamma_1 = \mathrm{Inf}(t)$ and player two's goal as $\gamma_2 = \neg\,\mathrm{Inf}(t) = \mathrm{Fin}(t)$. Thus, the game looks like the structure shown in Figure 5.2.

Now, consider the strategy profile $\vec{\sigma}$ where player 1 casts a vote of 1 voting credit for a and player 2 casts a vote of 1 voting credit for b at each stage of the game. It should be

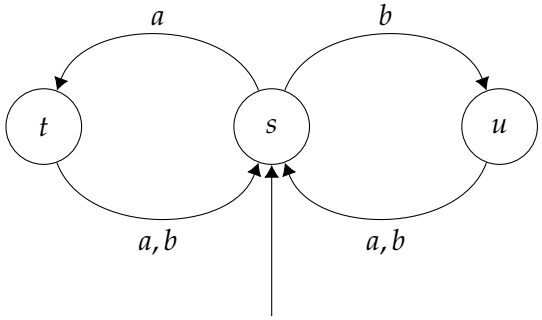

Figure 5.2: A game where one player's goal is the negation of the other, with a tree which models neither player's goal.

clear that from states t and u, the game will always move back to s, whilst in s, the game will non-deterministically move to one of t and u. Thus, the tree induced by this strategy profile, $\tau = \tau(\vec{\sigma})$ has a branch π_1 which alternates between s and t; i.e. $\pi_1 = ststst$ ..... But it also has a branch π_2 which alternates between s and u: $\pi_2 = sususu$ ..... We have $\pi_1 \vDash \gamma_1$ and $\pi_1 \nvDash \gamma_2$, as well as $\pi_2 \vDash \gamma_2$ and $\pi_2 \nvDash \gamma_1$. Putting this together, we see that $\tau \nvDash_\forall \gamma_1$ and $\tau \nvDash_\forall \gamma_2$. So we have a game where one player's goal is the negation of the other's, with a tree which models neither player's goals.

Thus, we need to formally define the notion of two-player, zero-sum games. Structurally, they are almost identical to iterated voting games instantiated with two players, where one player's goal is the negation of the other's, but they have different semantics. A two-player, zero-sum iterated voting game is a tuple,

$$G = \left(\mathrm{TS}, c_1, c_2, \gamma\right)$$

where,

- TS is a transition system;
- c_1 and c_2 are positive, integral amounts of *voting credits* for players 1 and 2 respectively;

- γ is an Emerson-Lei condition over St(TS).

These games are played as above: each player i casts a vote, giving a vote vector $v = (v_1, v_2)$. An action is then non-deterministically chosen from the winning actions of the vote, the game moves according to the transition relation, and this process repeats forever. Player 1 wins if the induced tree universally models γ, and player 2 wins otherwise.

Strategies are defined as before, but now that we are in the two-player, zero-sum setting, we (temporarily) do not have to concern ourselves with solution concepts; we just need to ask about *winning strategies*. A winning strategy for player 1 from a state s is a strategy σ_1 such that for all strategies σ_2, we have that

$$\tau((\sigma_1, \sigma_2), s) \vDash_\forall \gamma.$$

Similarly, a winning strategy for player 2 from a state s is a strategy σ_2 such that for all strategies of player 1 σ_1, we have that

$$\tau((\sigma_1, \sigma_2), s) \vDash_\exists \neg\gamma.$$

We say that a game is *determined* if every state of the game has a winning strategy for one of the players. Now, in these two-player, zero-sum games, we would like to determine the states from which player 1 and player 2 have a winning strategy. It is well-known that for deterministic, two-player, turn-based Emerson-Lei games [121, 152], this can be done in **PSPACE**. Moreover, it is also well-known that these games are determined. Ideally, we would now like to show that these results carry across to the non-deterministic, concurrent, voting setting. However, we will see that due to our non-determinism being 'balanced' (*i.e.* if a vote is split between multiple actions, then all those actions are non-deterministically taken), that our games have a far simpler characterisation, and that whether a player has a winning strategy from a given state is equivalent to deciding the

satisfiability or validity of the goal of the game. This was certainly a surprise to the authors (*i.e.* me), and raises a number of questions about the place of voting, and more generally, non-determinism in concurrent games. We leave any further discussion of this nature to the conclusion, and just focus on establishing our technical results for the remainder of the chapter.

We derive our result by splitting the analysis into a number of cases. First we note that we have formulated our games so that the players may have different numbers of voting credits. With multiple players, this feels like a reasonable assumption to make (for instance, under the law of some countries, companies have a 'one vote per share' system). However, in the two-player case, it may seem more unusual: if one player has more votes than the other, surely they can dictate what actions are taken within the system? This is exactly the case, and we codify this in the following lemma:

Lemma 9. *Let G be a two-player, zero-sum game such that $c_1 > c_2$ and let s be a state. Then player 1 has a winning strategy from s in G if and only if γ is satisfiable over TS(G) from s, and player 2 has a winning strategy from s in G if and only if γ is not satisfiable over TS(G) from s.*

Conversely, if $c_1 < c_2$, then player 1 has a winning strategy in G from s if and only γ is valid over G from s, and player 2 has a winning strategy in G from s if and only if γ is not valid over TS(G) from s.

Proof. We prove the first part of the first statement (the proofs of the remaining statements are similar). First suppose player 1 has a winning strategy in G from s; this means that they have a strategy σ_1 such that for all strategies σ_2, we have that $\tau((\sigma_1, \sigma_2), s) \vDash \gamma$. That means that for all branches π of $\tau((\sigma_1, \sigma_2), s)$, we have that $\pi \vDash \gamma$. Moreover, by definition, π is consistent with TS and starts from s, so we have that γ is satisfiable over TS(G) from s.

Now suppose that γ is satisfiable over TS(G) from s. So there exists some path π, consistent with TS, and starting from s, such that $\pi \vDash \gamma$. Consider the strategy for player

1 where at each time step k, they allocate all their voting credits to the action $\text{ac}[k]$ such that $\pi[k+1] = \text{tr}(\pi[k], \text{ac}[k])$. Since $c_1 > c_2$, no matter what strategy player 2 uses, the set of winning actions at each timestep will solely consist of the action chosen by player 1, and so the game will follow these choices, and hence, will follow π. Thus, player 1 has a strategy σ_1, such that for all strategies σ_2, we have that $\tau((\sigma_1, \sigma_2), s)$ has one branch, π, such that $\pi \vDash \gamma$; this means that player 1 has a winning strategy from s in G. □

This leaves us with the case of when $c_1 = c_2$, *i.e.* when the two players have equal voting power. Whilst one might intuitively expect this to be the 'interesting' case, we show that, surprisingly, these games also have a simple characterisation:

Lemma 10. *Let G be a two-player, zero-sum game such that $c_1 = c_2$, and let s be a state. The player 1 has a winning strategy from s in G if and only if γ is valid over $TS(G)$ from s. Similarly, player 2 has a winning strategy from s in G if and only if γ is not valid over $TS(G)$ from s.*

Proof. We prove the first statement — the proof of the second statement is similar. If the two players have the same number of voting credits, then we note that player 2 can force the existence of any branch consistent with $\text{TS}(G)$ and starting from s. As in the previous lemma, they can do this by simply allocating all their votes to the actions which induce the path. Then there are two options: if player 1 plays a vote which consists of splitting their voting credits among multiple actions, or not allocating all their credits, then the winning actions of the resulting voting profile will consist of just the action that player 2 has chosen. Alternatively, if player 1 plays a vote where they put all the voting credits into one action, then the set of winning actions will consist of the action that player 1 has chosen, along with the action player 2 has chosen (as they have an equal number of voting credits). Either way, at each time step, play will branch in a direction of player 2's choice. So given that a winning strategy for player 1 needs to make them a winner regardless of what player 2 does, and given that player 2 can make any consistent path appear as a branch when faced with any winning strategy of player 1, it follows that if player 1 has a

winning strategy, then every possible path consistent with TS(G) starting from s satisfies γ — in other words, that γ is valid over TS(G from s.

For the reverse direction, note that if γ is valid over TS(G) from s, then player 1 wins regardless of what actions are taken by each player, so any strategy is a winning strategy. □

Putting the above two lemmas together leads us to the following theorem:

Theorem 17. *Let G be a two-player, zero-sum game. Then it is determined. Moreover, determining if player 1 or player 2 has a winning strategy from a state s can be done in* **NP** ∪ **co-NP**.

Proof. For determinacy, first note that the assumptions on the voting credits among Lemmas 9 and 10 are exhaustive, and all possible games will be covered by one of the lemmas. Further note that the conditions for players 1 and 2 to be a winner from a given state under each assumption are negations of one another. Putting all this together, we may conclude that two-player, zero-sum, games are determined.

If $c_1 > c_2$, then by Lemma 9, determining if player 1 has a winning strategy from s is equivalent to determining if γ is satisfiable over TS(G) from s. By Theorem 16, we can determine if γ is satisfiable over TS(G) in **NP**. If $c_1 \leq c_2$, then by Lemmas 9 and 10, determining if player 1 has a winning strategy from s in G is equivalent to determining if γ is valid over TS(G) from s. Again, by Theorem 16, we can determine if γ is valid over TS(G) in **co-NP**. Putting this together, we see that we can determine if player 1 has a winning strategy from a given state in a two-player, zero-sum game can be done in **NP** ∪ **co-NP**.

In an identical fashion, we can reach the same conclusion for player 2. □

5.6 Iterated voting games revisited

With the two-player, zero-sum case decided, we are now in a position to turn our full attention to the multiplayer setting.

Noncooperative games

As always, in the non-cooperative case, the fundamental question we would like to understand is the following:

> E-NASH:
> *Given*: An IVG G, and an Emerson-Lei condition α.
> *Question*: Does there exist some Nash equilibrium $\vec{\sigma}$ such that $\rho(\vec{\sigma}) \vDash \varphi$ holds?

Before tackling the E-NASH problem, we will need to form a logical characterisation of the space of Nash equilibria. We say that a state, s, is punishing for a player i if there exists some strategy $\vec{\sigma}_{-i}$ such that for all strategies σ_i, we have that $\tau((\vec{\sigma}_{-i}, \sigma_i), s) \nvDash \gamma_i$. Denoting the set of punishing states for a player i by $\text{Pun}_i(G)$, we then say that a (state, vote) pair, $(s, \vec{v})$ is *punishing-secure* for a player i if for all votes v_i', there exists an action $\text{ac} \in \mathscr{W}(\vec{v}_{-i}, v_i',)$, such that we have $\text{tr}(s, \text{ac}) \in \text{Pun}_i(G)$. Informally, a (state, vote) pair is punishing-secure for a player if when that player deviates from a voting profile, then there is some path on which they can be punished by the remaining players.

Whilst the above concepts seem natural, one may worry how computable they are. We take a small step to allaying the reader's fears with an upcoming lemma, but for this, we need another definition: given a game G, we define the *punishment game* of player i with respect to G to be a two-player, zero-sum game,

$$G_i = (\text{TS}', c_1', c_2', \gamma'),$$

where,

- $\text{TS}' = \text{TS}(G)$;
- $c_1' = c_i(G)$;
- $c_2' = \sum_{j \in \text{Ag}(G) \setminus i} c_j(G)$;

- $\gamma' = \gamma_i(G)$.

Informally, the punishment game of player i explores what that player can achieve when all the other players 'team up' to act adversarially against them. As the name suggests, we can use punishment games to characterise punishing states, as we demonstrate with the following lemma:

Lemma 11. *A state s is punishing for a player i in G if and only if $Ag \setminus i$ has a winning strategy from s in the punishment game G_i.*

Proof. Trivial; follows by expanding definitions out. □

With this machinery in place, we are ready for the following technical lemma, which characterises the Nash equilibria of a game in terms of punishing states and punishment security:

Lemma 12. *Let G be an iterated voting game, and let τ_v be a tree of votes, with τ its associated tree of states. Then there exists some Nash equilibrium $\vec{\sigma} \in NE(G)$ with $\tau(\vec{\sigma}) = \tau$ if and only if for every loser i, there exists some branch $\vec{v}$ of τ_v, with a corresponding run of states π, such that for every $k \in \mathbb{N}$, we have that $(\pi[k], \vec{v}[k])$ is punishing secure.*

Proof. Let $\vec{\sigma}$ be a Nash equilibrium such that $\tau(\vec{\sigma}) = \tau$, and suppose the second part of the statement does not hold. So there exists some loser i, such that for all branches $\vec{v}$ of τ_v, with π the run corresponding to $\vec{v}$, there exists some $k \in \mathbb{N}$ such that $(\pi[k], \vec{v}[k])$ is not punishing secure for i. This means that there exists a vote v_i' such that for all actions $\mathrm{ac} \in \mathscr{W}(\vec{v}[k]_{-i}, v_i', \pi[k])$, we have $\mathrm{tr}(\pi[k], \mathrm{ac}) \notin \mathrm{Pun}_i(G)$. Now, consider the strategy σ_i' for player i, which consists of playing σ_i until play reaches the non-punishing-secure (state, action) profile in each branch, and then playing the v_i' as described in the previous sentence, before playing the winning strategy from the punishment game G_i. That means that player i has some strategy σ_i' such that on all branches π of $\tau(\vec{\sigma}_{-i}, \sigma_i')$ we have $\pi \vDash \gamma_i$ (by construction). In turn this implies that $\tau(\vec{\sigma}_{-i}, \sigma_i') \vDash_\forall \gamma_i$, contradicting that $\vec{\sigma}$ is a Nash

equilibrium. Thus, it must be the case that for every loser i, there exists some branch $\vec{v}$ of τ_v, with corresponding run π, such that for every $k \in \mathbb{N}$, we have that $(\pi[k], \vec{v}[k])$ is punishing secure.

For the other direction, suppose τ is a tree with the property that for every branch $\vec{v}$ of τ_v, with corresponding run π, and every $k \in \mathbb{N}$, we have that $(\pi[k], \vec{v}[k])$ is punishing secure for every loser j. Then there exists a Nash equilibrium $\vec{\sigma}$ with $\vec{\sigma} = \tau$ which looks like the following: the players coordinate their votes to follow τ_v. If any loser i deviates from following τ_v on the branch which is punishing-secure for every (state, action) pair, then the remaining players play their winning strategy from the punishment game for i, ensuring that player i does not get their goal achieved. Thus, we have formed a strategy profile $\vec{\sigma}$ such that $\tau = \tau(\vec{\sigma})$ with the property that no loser has a unilateral deviation to a winning strategy; that is, a Nash equilibrium. □

With these lemmas in place, we are ready to attack the E-Nash problem, using conceptually similar techniques to those used in the proof of Theorem 1:

Theorem 18. *The E-Nash problem is in* **NP** ∪ **co-NP**.

Proof. Let G be a game, and let α be an Emerson-Lei condition. Now, consider the following algorithm: first, we calculate the set of punishing states for each player. Then we guess a coalition of winners $W \subseteq \mathrm{Ag}$. Having done this, consider the graph, $\mathscr{G}$, induced by the underlying transition system of the game. We prune this graph to remove those transitions which are not punishing secure for at least one loser. Having done this, we also guess a permset X which satisfies the Emerson-Lei condition, and remove any states that are not in X. Now, we claim that there is a Nash equilibrium in G which models a specification α if and only if $\mathscr{G}$ has a loop, $\mathscr{C}$, which visits every state in X, and is reachable from the start state of the game.

Suppose G has a Nash equilibrium, $\vec{\sigma}$ which models α. By Lemma 12, we know that there exists some tree of votes, τ_v with the property that for every loser i, there exists

some branch $\vec{v}$ of τ_v, with a corresponding run of states π, such that for every $k \in \mathbb{N}$, we have that $(\pi[k], \vec{v}[k])$ is punishing secure. Now, pick any such $\vec{v}$, and note that since $\pi \vDash \alpha$, there will exist two indices, j, k, with $\pi[j] = \pi[k]$, such that every state of X is accounted for between $\pi[j]$ and $\pi[k]$. Additionally (assuming we guessed X correctly), we may assume that no state outside X lies within $\pi[j]$ and $\pi[k]$. We claim that the loop $\mathscr{C} = \overline{\pi[j..k]}$ has the desired properties. First note that each (state, action) pair induced by $\vec{v}$ is punishing secure for at least one player i, and that $\pi[j..k]$ only contains states within X, so it follows that $\pi[j..k]$, and therefore $\mathscr{C}$ lies within $\mathscr{G}$. Furthermore, by construction, $\mathscr{C}$ visits every state in X and is reachable from the start state of the game.

Conversely, suppose that $\mathscr{G}$ has a loop $\mathscr{C}$ which visits every state in X and is reachable from the start state of the game. So consider the strategy $\vec{\sigma}$, where the players first coordinate to get to $\mathscr{C}$, and then follow it repeatedly. First we note that $\vec{\sigma}$ does not split votes across multiple actions at any stage, and so induces a single-branched tree of actions, which we denote $\vec{v}$, and its corresponding run of states π. By construction of the graph, we have that $(\pi[k], \vec{v}[k])$ is punishing-secure for all $k \in \mathbb{N}$, so if any loser deviates from $\vec{\sigma}$ at timestep k with a vote v_i', then there exists some action $\mathrm{ac} \in \mathscr{W}((\vec{v}[k]_{-i}, v_i',)$ such that $\mathrm{tr}(s, \mathrm{ac}) = s' \in \mathrm{Pun}_i(G)$. So we augment $\vec{\sigma}$ such that if play enters s', then the remaining players will play the appropriate punishing strategy from the punishment game G_i. Thus, $\vec{\sigma}$ is a Nash equilibrium by construction. It just remains to show that $\tau(\vec{\sigma})$ models α. To see this, note that any run with the permset X models α, and also that by construction we have that $\mathrm{permset}(\pi) = \mathrm{permset}\left(\overline{\mathscr{C}}\right) = X$; it follows that $\pi \vDash \alpha$. Thus, all branches of the tree induced by $\vec{\sigma}$ model the specification; now we just need to show that it is indeed a Nash equilibrium.

Thus, to decide if a provided instance, (G, α), is a member of E-Nash, we form $\mathscr{G}$, and determine if it has a loop with the described properties. If it does, then we accept, otherwise we reject.

In terms of complexities, by Lemma 11 and Theorem 17, calculating the punishing

states for each player can be done in **NP** ∪ **co**-**NP**. We also non-deterministically guess a winning coalition, $W \subseteq \mathrm{Ag}$ and a permset X concurrently, as all of these operations can be independently of one another. Pruning the graph can be done in polynomial time, as can determining if $\mathscr{G}$ has a loop which visits every state in X — this latter step can be done by determining if the graph forms one strongly-connected component with every state of X within it; this can be done with Tarjan's algorithm [200]. □

Cooperative games

In the cooperative setting, we will consider the following decision problems:

> Coalition-SAT:
>
> *Given*: An IVG G, a coalition C, and an Emerson-Lei condition α.
>
> *Question*: Does there exist a strategy vector $\vec{\sigma}_C$ such that $\rho(\vec{\sigma}_C, \vec{\sigma}_{\mathrm{Ag}\setminus C}) \vDash \alpha$, for all responses $\vec{\sigma}_{\mathrm{Ag}\setminus C}$?

> E-Core:
>
> *Given*: An IVG G, and an Emerson-Lei condition α.
>
> *Question*: Does there exist some member of the core $\vec{\sigma}$, such that $\rho(\vec{\sigma}) \vDash \alpha$?

Theorem 19. *The Coalition-SAT problem for IVGs lies in* **NP** ∪ **co**-**NP**.

Proof. Given a game G, a coalition C and an Emerson-Lei condition α, form a two-player zero-sum game

$$G' = (\mathrm{TS}', c_1', c_2', \gamma'),$$

where,

- $\mathrm{TS}' = \mathrm{TS}(G)$;
- $c_1' = \sum_{i \in C} c_i(G)$;
- $c_2' = \sum_{i \in \mathrm{Ag}(G)\setminus C} c_i(G)$;

- $\gamma' = \alpha$.

Note that this construction can be done in polynomial time. We then ask if player 1 has a winning strategy in G' from the state s^0. By Theorem 17, we can determine this in **NP** $\cup$ **co-NP**. We then note that player 1 has a winning strategy in G' from the state s^0 if and only if the coalition C has a strategy $\vec{\sigma}_C$ such that for all counter-strategies $\vec{\sigma}_{\mathrm{Ag}\setminus C}$, we have $\tau(\vec{\sigma}_C, \vec{\sigma}_{\mathrm{Ag}\setminus C}) \vDash \alpha$ in G — this follows by construction. Thus, we may conclude that membership for Coalition-SAT can be done in **NP** $\cup$ **co-NP**. □

Ideally, we would like to perform the inverse construction, and reduce from the two-player, zero-sum setting to the multiplayer setting, in order to yield some sort of hardness result. However, as discussed earlier (and illustrated in Example 8), the multiplayer model cannot capture player 2's universal goal. Whilst we do not provide the lower bound, we will try to work around this deficiency of our model in Section 5.7.

With this result in place, we can now turn our attention to the E-Core problem:

Theorem 20. *The E-Core problem for IVGs lies in* Σ_2^P.

Proof. We perform the following algorithm: first guess a coalition of winners, $W \subseteq \mathrm{Ag}$, and check that the grand coalition has some strategy which models the formula,

$$\bigwedge_{i \in W} \gamma_i \wedge \bigwedge_{i \in \mathrm{Ag}\setminus W} \neg\gamma_i \wedge \alpha. \tag{5.1}$$

If the grand coalition does have such a strategy, then we check that no coalition of losers, $L \subseteq \mathrm{Ag} \setminus W$, can beneficially deviate; this equates to checking if the coalition L can force the formula,

$$\bigwedge_{j \in L} \gamma_j \tag{5.2}$$

to be true. If no such coalition of losers can beneficially deviate, then we have found a strategy profile which models the specification, from which no coalition of losers can

beneficially deviate — that is, a member of E-Core. If instead some coalition of losers can beneficially deviate, we turn our attention to the next coalition of possible winners. We also do this if the grand coalition does not have a strategy that models both the current winners' goals, the specification, and the negations of the losers' goals. Finally, if we have exhausted all possible coalitions of winners in the above fashion, then the provided game and specification is not a member of E-Core, and we reject.

Note that this can be done in Σ_2^P: first, we non-deterministically guess the coalition of winners. We then use an **NP** oracle to check that Ag can achieve 5.1 (this can be done as Coalition-SAT is in **NP** $\cup$ **co-NP**). We follow this by a call to a **co-NP** oracle to show that there is no coalition $L \subseteq \text{Ag} \setminus W$, that can achieve 5.2 (again, as Coalition-SAT is in **NP** $\cup$ **co-NP**, its complement is also in **NP** $\cup$ **co-NP**). □

5.7 Tree-based goals

Up until now, our players have had goals which were path based, but interpreted relative to trees. Whilst this is an intrinsically interesting game model in its own right, it has a key flaw: as existential satisfaction cannot be encoded in the goals of our players, we cannot readily derive the lower bounds to match the upper bounds on our decision problems. Moreover, universal satisfaction was an arbitrary choice.

In this section, we offer a (very) brief discussion of an extension to our work, namely Emerson-Lei condition defined over trees, baking in universal or existential satisfaction directly into the goal. Tree-based Emerson-Lei conditions are given by the following grammar:

$$\alpha ::= \neg\alpha \mid \exists\alpha'$$
$$\alpha' ::= \text{Inf}(s) \mid \neg\alpha' \mid \alpha' \vee \alpha',$$

As one might expect, we identify $\neg\exists\alpha'$ with $\forall\alpha'$, and use the usual shorthands and

rules of propositional logic for the remaining connectives. So with tree-based Emerson-Lei conditions, we can write expressions such as $\forall(\text{Inf}(s) \vee \text{Fin}(t))$ or $\exists(\text{Inf}(s) \wedge \text{Inf}(t))$, but not expressions like $\exists\forall\ \text{Inf}(s)$ or $(\forall\ \text{Inf}(s)) \wedge (\exists\ \text{Fin}(t))$. Their semantics are defined in the natural way, relative to state-labelled trees, τ:

- If $\alpha = \neg\beta$, then $\tau \vDash \alpha$ if and only if it is not the case that $\tau \vDash \beta$;
- If $\alpha = \exists\alpha'$, then $\tau \vDash \alpha$ if and only if there exists some path π of τ such that $\pi \vDash \alpha'$.

Note that in the second bullet above, we are talking about two different modes of satisfaction: the semantics of tree-based Emerson-Lei conditions are defined in terms of the semantics of path-based (*i.e.* the usual) Emerson-Lei conditions.

We can now define a new game model, where the players' goals are given by tree-based Emerson-Lei conditions. Formally, a tree-based iterated voting game takes the same form as a (regular) iterated voting game, except for each player i, their goal γ_i is given by a tree-based Emerson-Lei condition. The semantics of these games is as before, except that we interpret players' goals relative to trees in the way discussed above.

With this in place, we can derive a number of results relatively easily:

Theorem 21. *Let G be a tree-based game with two players $\{1, 2\}$ such that $\gamma_2 = \neg\gamma_1$. Then the problem of determining if player 1 has a winning strategy from a given state is in* **NP** $\cup$ **co**-**NP**.

Proof. Directly maps to the setting of two-player, zero-sum Emerson-Lei games. □

Using this, we can also understand the E-Nash and Coalition-SAT problems in a similar manner to before:

Theorem 22. *For tree-based voting games, the E-Nash problem is in* **NP** $\cup$ **co**-**NP**.

Proof. Proof similar to before; one just needs to appropriately adapt the notion of punishing states. □

And in the cooperative setting:

Theorem 23. *For tree-based voting games, the* Coalition*-SAT problem is in* **NP** ∪ **co**-**NP**.

Proof. Proof similar to before. □

Now, we would also like to analyse the core in our new type of games, but we quickly run into a problem: the usual techniques for reasoning about beneficial deviations usually amount to determining if some coalition has a winning strategy for the conjunction of their goals. However, we cannot express conjunctions in our framework — one player's goal may be of the form $\exists \alpha$ and another's of the form $\forall \beta$.

To obtain full generality, we would have liked to allow conjunction and disjunction over the path quantifiers; that is, it would be great if we could write expressions like $\forall \operatorname{Inf}(s) \wedge \exists \operatorname{Fin}(t)$. However, whilst it is clear how we might extend two-player, zero-sum games to accommodate this, it is not obvious how we might adapt our existing proof techniques (such as those used in the proof of Theorem 17 or those used in McNaughton's original proof [152]) to provide the analogous results in this new setting. For example, given that we cannot simply label players as existential or universal, it becomes unclear how we should generalise even the simplest definitions, such as what it means for a player to be able to force play to visit a set of states (an important concept in McNaughton's proof). Whilst in this thesis we have taken some healthy preliminary steps in handling tree-based Emerson-Lei goals, it is clear that further work is needed to fully characterise and understand them in full generality.

Chapter 6

Conclusion

I finish by summarising my results, and more informally discussing the work undertaken. I may not offer a full discussion of everything we covered, but instead, I will talk about the parts of this thesis which preoccupied my thoughts; what I feel I might have done differently if I knew earlier what I know now; and my ideas for future work. Each section will be accompanied by a table featuring the complexity results from the corresponding chapters. I will then discuss a few more possibilities for future work which fall under the umbrella of rational verification, before I leave you with a short-and-sweet sign-off.

6.1 Themes on mean-payoff games

We used Emerson-Lei conditions as a natural, expressive, computationally tractable way of reasoning qualitatively about concurrent games. In particular, we established several results within the rational verification framework, providing logical characterisations of the Nash equilibrium and the core in our setting, along with complexity results for the rational verification decision problems for both the Nash equilibrium and the core over concurrent game structures, as well as the Nash equilibrium over a succinct representation of our concurrent games.

My main takeaway from this topic has been that the core in the mean-payoff setting is an unexpectedly tricky nut to crack. Specifically, even the most simple complexity bounds are tantalisingly out of reach. This is in stark contrast to the LTL setting [103], where the core has a relatively straightforward, direct characterisation in terms of coalitional power, which we can then translate into effectively one large ATL* formula. These techniques do not generalise to the mean-payoff setting, and no alternative route seems to readily offer itself up. In terms of exactly why this problem has proven to be so stubborn, I can offer a few intuitive explanations which might give you a feeling of where the difficulty lies:

- Two-player, zero-sum, multi-mean-payoff games require infinite memory for player 1 to act optimally [207]. This could imply that members of the core may need infinite memory, which in turn may mean that player payoffs are arbitrary real numbers or are 'big' rationals, which are computationally hard, or even impossible, to guess, taking us out of the polynomial hierarchy (which is where most of the proof techniques that I am aware of lie with respect to these types of games);

- As we highlighted at the end of the cooperative portion of this chapter (in Lemma 7 and the surrounding text), there is a natural link between the core in mean-payoff games and the core in the corresponding NTU games. However, necessary and sufficient conditions for core non-emptiness tend to be rather (dare I say) *technical* and *complicated* [40, 46, 180, 189, 190, 191]. Moreover, many of the arguments in these papers have a geometric flavour to them — this gives me the feeling of being a computer scientist with a square peg and a round hole. Whilst some key insights have already been obtained using geometric techniques in work such as Brenguier and Raskin [2015], I feel this an area which could benefit from being further analysed with such ideas.

- Having non-dichotomous preferences, in the form of mean-payoff goals, in contrast to the dichotomous preferences of LTL goals, is a double-edged sword. Whilst you

do gain access to techniques such as linear programming, and can take advantage of the structure of the real numbers, along with the geometry of the sets of payoffs that players can achieve, you lose a lot of the nice features of dichotomous preferences. In particular, you lose core non-emptiness, as 'cycles' of strategies, linked by beneficial deviations, become possible, and you also lose natural definitions, such as *fulfilled coalitions*. Yes, we did generalise this in the form of *lower bounds*, but unfortunately, the results did not have the good manners to generalise across with them.

Overall, the message you should glean from the above is the following: the core in mean-payoff games is very different from the core in LTL games, or any sort of games with dichotomous preferences. If we are to understand this solution concept better in such a scenario, it is almost inevitable that it will require the development of a brand new, orthogonal set of proof techniques.

In addition to the core, the other most obvious open problem is closing the complexity gap for the WRMG-E-Nash problem. Following this, there are other directions that appear to be fruitful, as well as interesting and worthwhile — for example, introducing both imperfect information and non-determinism offers a closer approximation to real-world systems, as well as raising interesting mathematical questions (although, given the results of Chapter 5, I am rather wary of non-determinism now).

The work from Chapter 3 has been published under the title 'Mean-payoff games with ω-regular specifications' [105, 196], but as one reviewer helpfully pointed out (albeit, a bit too late), this is a bit of a misnomer: not all ω-regular conditions can be expressed in the Emerson-Lei formalism. Specifically, whilst you can express the idea of states being visited infinitely or finitely often with Emerson-Lei conditions, you cannot do anything more granular than this, such as specifying prefixes of a run, or supplying an order to the states visited infinitely often. To this end, I think exploring games where the player goals, or the system specification, are given by an arbitrary ω-regular automata is a natural, rich, yet (seemingly) unexplored vein. Moreover, I believe that many of the techniques set out

Thm. #	Decision problem	Complexity
1	E-Nash	**NP**-complete
2	∃-Ben-Dev with memoryless strategies	**NP**-complete
3	Core-Membership with memoryless strategies	**co-NP**-complete
4	E-Core with memoryless strategies	In Σ_2^P
4	A-Core with memoryless strategies	In Π_2^P
5	Lower-Bound	**co-NP**-complete
6	WRMG-E-Nash	In **NEXPTIME** and **EXPTIME**-hard

Table 6.1: Summary of complexity results for Chapter 3.

in my work would readily generalise to this setting.

6.2 Iterated allocation games

We introduced iterated allocation games as a way of modelling resource-bounded multi-agent systems, before instantiating the decision problems of rational verification with respect to these games, establishing their complexities, and exploring variations of the basic model. This included considering a subset of LTL which lends itself to our model, as well as introducing agents with more nuanced behaviours — specifically, resource-bounded agents who want to satisfy an LTL property, whilst also minimising resource usage. The main results of the chapter extend previous work on iterated games as well as on rational verification, in both the non-cooperative and cooperative game settings where either Nash equilibrium or the core of a game are taken to be the desired outcomes of the game.

The inherent 'monotonicity' of iterated allocation games suggests an extended exploration of the considered decision problems with monotone LTL goals could be fruitful. However, if I were to start from scratch with this piece of work, I would eschew LTL entirely, and use Emerson-Lei conditions instead. The **2EXPTIME**-completeness of LTL synthesis has a tendency to 'hide' the interesting parts of what is going on in an argument by allowing you to do exponential operations without affecting the overall result.

Thm. #	Decision problem	Complexity
7	IAS-Model-Checking with finite-memory input	In **PSPACE**
8	Nash-Membership with finite-memory input	**PSPACE**-complete
9	Beneficial-Deviation with finite-memory input	**PSPACE**-complete
10	E-Nash	**2EXPTIME**-complete
10	Core-Membership with finite-memory input	**2EXPTIME**-complete
10	Coalition-SAT	**2EXPTIME**-complete
10	E-Core	**2EXPTIME**-complete
11	Monotone-Coalition-SAT	In **PSPACE**
12	Necessary-Resource	**2EXPTIME**-complete
13	Resource-Bound-SAT	**2EXPTIME**-complete
14	E-FSNE$^{\varepsilon}$	**2EXPTIME**-complete

Table 6.2: Summary of complexity results for Chapter 4.

One major qualm that some reviewers of this work had was that it looks like there should be a polynomial translation from IAGs to IBGs (which would make IAGs 'just' an interesting application of the IBG framework, rather than a framework worth studying in its own right). I have tried to address those concerns head-on in the thesis, specifically in the text following Theorem 10. To recap, I am strongly convinced that there is no such polynomial translation, as any attempt to map IAGs to IBGs appears to yield an exponential blowup somewhere; if you manage to remove it in one place, it reappears in another. Whilst a formal proof of this has eluded me, I believe that studying these games with Emerson-Lei conditions could offer a solution — given that Emerson-Lei synthesis is **PSPACE**-complete, games with goals of this form might be the perfect candidate for showing that some sort of exponential blowup is necessary when reasoning about iterated allocation games. Would the complexity of problems such as E-Nash in this setting match the **PSPACE** bound, or is the exponential blowup argument of Lemma 8 truly necessary, pushing the complexity up to **EXPTIME**? These tantalising questions will have to wait for future work.

6.3 Voting games

We introduced voting games as a mechanism for reaching consensus, and settling disagreements among groups of agents. After discussing the one-shot case for motivational purposes, we generalised this to the iterated setting, where we established a number of results. This included developing an algorithm for computing the winners in a two-player, zero-sum game, and determining its computational complexity; analysing the usual decision problems of rational verification with respect to voting games in both the non-cooperative and cooperative settings; and generalising the model further to include agents with tree-based goals. Given the multiple novel elements in our framework, such as the Emerson-Lei conditions in rational verification, along with voting and the non-determinism it induces, these results necessitated the need for new proof techniques and arguments. Despite this, our framework offers a more computationally tractable view of rational verification, whilst still being a very expressive formalism.

The fact that determining the winner in a two-player, zero-sum voting game reduces to checking satisfiability or validity of the player 1's goal over the game structure was a shock to me. A priori, I expected that this framework would be a generalisation of McNaughton's original formulation [152], and that we would obtain (at least) a **PSPACE**-completeness result by modifying the original argument. However, the three core differences between our frameworks and McNaughton's (namely, the concurrency, the voting and the non-determinism of our games), led to a significant divergence in semantics — large enough to almost trivialise the resulting games.

As hinted throughout Chapter 5, this problem comes about because the non-determinism is 'balanced': if two actions have an equal number of votes, then the game will split in those two directions. I see two ways out of this predicament.

The first is to change the players' goals; universal satisfaction is a really strong condition to have on a tree, whilst existential satisfaction is quite weak. Perhaps the tree-based

Emerson-Lei conditions described at the end of Chapter 5 (*i.e.* formulae like $\exists \operatorname{Inf}(s) \vee \forall \operatorname{Fin}(t)$) would hit that 'sweet spot' in terms of expressiveness and strength? However, as stated in the text, I believe this would require the development of a new set of proof techniques.

The second idea we could try is to question the role of voting in our games: it is clear that any 'sensible' voting mechanism will induce the balanced non-determinism described. Maybe we should scrap voting entirely, and replace it with a function: one that takes a signal from each player, and spits out a set of actions that should be taken non-deterministically. It should be clear that this subsumes voting, and captures a wider range of scenarios, whilst also generalising beyond balanced non-determinism. However, intuitively, I question the applicability of such a model: perhaps such a function could be interpreted as the system designer or controller imposing their will on the world? Additionally, it is hard to say in advance what results might drop out of such a framework. One thing is clear however: Chapter 5 has raised a lot of questions, and we must be selective in choosing which we wish to answer.

There are two more speculative options for refining the game model to make it more strategically interesting whilst retaining non-determinism[1]. One would be to introduce a player whose role is to pick between tied actions. Whilst this option initially sounds attractive, I feel there are a number of difficulties with formalising this idea. For instance, does this new player have any preference over runs? If they do not, then I feel that this idea brings no advantage over our (direct, interpretable) techniques for understanding these games, at the cost of introducing an extra player. If this new player does have some preference over runs, then what would the interpretation of such a player be? Are they like a system designer, or controller in the game? Is this a reasonable assumption for a model of a real-world system? Furthermore, implementing this idea either risks blowing up the size of the game structure, or becoming some sort of hybrid turn-based/concurrent

[1]Both initially suggested by Jean-François Raskin

game. Finally, I feel introducing an additional player could make it harder to appeal to existing techniques, and might even require novel lines of reasoning. Whilst I am not ruling out this technique for reasoning about non-determinism, I feel it very much has the potential to introduce more problems than it solves.

The second speculative option would be to make the number of voting credits available to each player dependent on the current state of the game. Again, I think this leads to a question of interpretation: why should players lose or gain voting power based on gameplay so far? If this were the case, then the votes look more like resources to me, and we are in a non-deterministic version of the sorts of models encountered in Chapter 4. However, interpretation concerns aside, this option could certainly inject a certain amount of strategic depth to our games, and would certainly be worth exploring in the future.

We should also take the time to compare our model to the *bidding games* [28] described in Chapter 1, as there are parallels between the two formalisms, and bidding games were certainly a key inspiration when we formulated our voting games. To recapitulate: bidding games are two-player, concurrent games on finite graphs, where the players each have a budget, and bid for the right to choose the next move. There are a number of different flavours of bidding game, which are largely differentiated by the method of deciding who receives the winning bid (such as the other player, or the 'bank'), as well as the winning condition for the game. Moreover, models of both discrete and continuous bidding have been studied in bidding games. We note the following key contrasts between bidding games and our voting games:

- Voting games are multi-player whilst bidding games are two-player; moreover, it is unclear how easily bidding games and the associated proof techniques would generalise to multiple players;

- In bidding games, players have varying 'power' to enact their well at as the game goes on (due to their increasing and decreasing budget), whilst players in our voting

games have constant 'power' throughout the game;

- Voting games are formulated in terms of discrete bidding whilst bidding games can be instantiated with either discrete or continuous bidding;
- We study the goals and specifications of our voting games with a non-deterministic lens, whilst to my knowledge, bidding games are typically studied with a deterministic lens;
- Both voting and bidding games have natural, real-world interpretations, but crucially, they have different interpretations (as offered up by their names).

We make one final observation about bidding games that this author found interesting: in the setting of discrete bidding, given the deterministic formulation, one needs to determine how to break ties when both players bid the same amount (this issue is allegedly not an issue in the context of continuous bidding [29]). Aghajohari, Avni and Henzinger [2021] showed that in general, even with a deterministic tie-breaking rule, these games may not be determined; given our surprising result for voting games, this author wonders if the price of determinacy in these voting/bidding-like settings is a strategic trivialisation of the game. With this having been said, Aghajohari, Avni and Henzinger [2021] does offer some classes of tie-breaking mechanisms and winning conditions which do allow determinacy, and it would be instructive to see if any insight from these results could be carried over to our framework.

Finally, one may note that there is a distinct lack of lower bounds in Chapter 5. This is due to the fact that whilst we know that solving two-player, zero-sum games can be done in $\mathbf{NP} \cup \mathbf{co}\text{-}\mathbf{NP}$, we have been unable to determine if this upper bound is optimal. Is our problem complete for $\mathbf{NP} \cup \mathbf{co}\text{-}\mathbf{NP}$ (of which I am not aware of any complete problems), or does it lie in some subset of $\mathbf{NP} \cup \mathbf{co}\text{-}\mathbf{NP}$? Certainly, given the problem's close relationship with satisfiability and validity, we should be able to readily derive **NP** and **co-NP** hardness results, in turn yielding those lower bounds for our other decision problems.

Thm. #	Decision problem	Complexity
15	E-Nash in one-shot games	Σ_2^P-complete
16	Emerson-Lei-SAT	**NP**-complete
16	Emerson-Lei-Tautology	**co-NP**-complete
17	Given a two-player, zero-sum iterated voting game, a player and a state, does that player have a winning strategy from that state?	In **NP** ∪ **co-NP**
18	E-Nash	In **NP** ∪ **co-NP**
19	Coalition-SAT	In **NP** ∪ **co-NP**
20	E-Core	In Σ_2^P
21	Given a two-player, zero-sum iterated voting game with tree-based goals, a player and a state, does that player have a winning strategy from that state?	In **NP** ∪ **co-NP**
22	E-Nash for tree-based games	In **NP** ∪ **co-NP**
23	Coalition-SAT for tree-based games	In **NP** ∪ **co-NP**

Table 6.3: Summary of complexity results for Chapter 5.

However, we have omitted any formal presentation of any such results, and leave this as another task for future work.

6.4 Other future work

With respect to rational verification, I am personally interested in exploring two further avenues: compositionality and tooling. What do I mean by compositionality? Thus far, beyond the particular requirements that a game model has, we have made no further assumptions about our games[2]. However, most real-world systems exhibit a strong structure; in particular, they may consist of independent, yet interacting parts, or they may behave in a predictable way that cannot be directly encoded in the game model. So, given a large game, can we break it up into a collection of smaller games and analyse them, before pooling that understanding into a holistic view of the original game? Or we can look at the dual problem: we have an understanding of how a set of small games behave — how

[2]The small disclaimer to this is that we did consider reactive module games in Chapter 3; one of the motivating factors for reactive modules is that they support compositional reasoning [19]. However, we did no such reasoning in the thesis, and did not use any compositional features of reactive modules in our results.

do they behave when they are somehow composed together? I believe we can utilise this structure to synthesise models of a specification more readily — decompose the system, analyse the subparts, then pool that understanding together to synthesise a model. This is a general idea, but one that I believe is particularly suited to the problem at hand: it seems to be a very natural direction in game theory and systems verification, but my personal intuition is that this would require a careful formulation, along with a lot of careful work, to yield useful results.

There has already been some preliminary work on this idea [11, 19, 70], and indeed, I briefly considered this problem myself as part of my MSc, looking at 'local equilibria' in games [107]. Reactive modules [19] offer the model and syntax to compose systems together, but to my knowledge, this feature has not been exploited in model checking beyond the MOCHA project [22]. Another idea is to use category theory to reason about compositionality, of which there has been a recent surge of activity [98], but this has the downside of involving the overhead of category theory. I would be interested in taking some time to explore these ideas further and to see if a more game-theoretic presentation could be yielded from these works.

In terms of tooling, I believe there is a lot of relatively 'low-hanging fruit' available. As mentioned in Chapter 1, our main software tools for rational verification are EVE [107] for LTL games and PRALINE [52] for multiplayer, mean-payoff games. However, both pieces of software are effectively 'proof-of-concepts', rather than hardened, production-ready implementations. This may feel like a harsh comment to make, but compare the situation to SMT solvers: Z3 [159] and CVC5 [35] are two popular open-source implementations. They are well-maintained, seeing continual updates from both the authors and the community; each have comprehensive test suites and well written documentation; and both can be accessed through a standard interface, namely the SMT-LIBv2 format [36] — this latter point is crucial, as it means you do not need to have any tool-specific knowledge; you can treat each tool as a black box for solving SMT formulae, allowing you to

easily embed different solvers into other applications, or compare their respective performance. As a result of these factors, these two projects have thousands of GitHub stars between them (a reasonable proxy measure of their popularity). I have singled out EVE and PRALINE, as these are the most relevant tools for rational verification, but in fact, almost all the software mentioned in the introduction (PRISM-GAMES being the notable exception), have been 'one-off' releases of code with, at most, very fleeting updates and little-to-no documentation. Clearly, there is a niche to be filled here.

So where can we start with this grand endeavour? In the same way that Babiak et al. [2015] provided a standard, formal syntax for specifying ω-automata, I feel it would be useful for there to be a standard syntax for concurrent games. This would encourage standardisation, and we could build on top of this, hopefully leading us to production-quality software for manipulating and solving concurrent games. This would dramatically aid research, helping to experimentally produce and verify conjectures. Appealing to the parallel of SMT solvers once more, my hope is that in the same way we have SAT [118], SMT [208] and program synthesis [23] competitions, that we might one day see competitions for synthesis in games and for rational verification: not only would this encourage serious thought about the encoding of real-world scenarios into these frameworks, but also the development of practical tools which may in turn yield interesting theoretical insights.

6.5 Concluding remarks

In this thesis, we introduced a number of quantitative extensions to the rational verification framework. In doing so, we have brought verification closer to modelling real-world systems, such as those with agents who have non-dichotomous preferences, and those with resource-bounded agents. The hope is that more systems of this nature will now be able to be expressed in the rational verification framework, making automated design and

synthesis of safe and efficient systems more feasible. Moreover, we have developed a new set of proof techniques which will hopefully not only allow further results to be yielded within rational verification, but will also aid the wider systems verification community, as well as the larger computer science community, in general.

Glossary

Acronyms

ATL/ATL*

: Alternating-time temporal logic.

BVG

: Boolean voting game.

CGS

: Concurrent game structure.

CTL

: Computation tree logic.

GR(1)

: Generalised reactivity(1).

IAG

: Iterated allocation game.

IAS

Iterated allocation structure.

IBG

Iterated Boolean game.

IVG

Iterated voting game.

LTL

Linear temporal logic.

NTU

Non-transferable utility.

RMIAG

Resource-minimisation iterated allocation game.

SRML

Simple reactive modules language.

WRMG

Weighted reactive module game.

Complexity classes

2EXPTIME

The class of decision problems which can be solved in doubly exponential time by a deterministic Turing machine.

3EXPTIME

The class of decision problems which can be solved in triply exponential time by a deterministic Turing machine.

co-NP

The class of decision problems which comprises the complements of problems in **NP**.

co-UP

The class of decision problems which comprises the complements of problems in **UP**.

EXPTIME

The class of decision problems which can be solved in exponential time by a deterministic Turing machine.

FPT

Informally, the class of decision problems which can solved in polynomial time when one of their parameters is held constant. For a formal definition, see [81].

NEXPTIME

The class of decision problems which can be solved in exponential time by a nondeterministic Turing machine.

NP

The class of decision problems which can be solved in polynomial time by a nondeterministic Turing machine.

Π_2^P

The class of decision problems which comprises the complements of problems in Σ_2^P.

PSPACE

The class of decision problems which can be solved using polynomial space by a deterministic Turing machine.

P

The class of decision problems which can be solved in polynomial time by a deterministic Turing machine.

Σ_2^P

The class of decision problems which can be solved in polynomial time by a non-deterministic Turing machine with an NP oracle.

TFNP

The class of total function problems which can be solved in polynomial time by a non-deterministic Turing machine.

UP

The class of decision problems which can be solved in polynomial time by an unambiguous, non-deterministic Turing machine; that is, problems with a unique certificate.

Notation

$|S|$

The size of the set S.

Ac_i

The actions available to player i in a concurrent game structure.

Ag

A set of agents.

N

An alternative syntax for the set of agents — generally encountered in iterated Boolean games and its relations.

A_i

The allocations available to player i in an iterated allocation structure.

A

An SRML arena; also used to denote the set of allocations over all the agents of an iterated allocation structure.

$\arg\max(f)$

The elements in the domain of f which attain its maximum value.

AP

A set of atomic propositions.

M

A concurrent game structure.

c_i

In Chapter 4, a cost function for player i, used to define player preferences in RMIAGs; in Chapter 5, the number of voting credits that player i possesses.

ctr(g)

The 'controlled' of the guarded command g — that is, the propositional variables that will be affected upon executing g.

α

An Emerson-Lei condition in Chapters 3 and 5; also used for allocations, assignments of player resources to atomic propositions, in Chapter 4.

$\varnothing$

The empty set.

enabled$_i(v)$

The guarded commands available to player i under the valuation v.

en

An endowment function, detailing how many resources each player is equipped with in an iterated allocation structure.

exec$_i(g, v)$

The valuation of player i's variables obtained by executing the guarded command g under the valuation v.

$\bot$

The logical formula, false.

Fin(s)

An atom in an Emerson-Lei condition, meaning 'the state s is only visited finitely often'.

G

A game — the type of which depends on the context.

γ_i

The goal of player i; the form of this goal depends on the setting.

guard(g)

The 'guard' of the guarded command g — that is, the precondition for taking the action defined by g.

Inf(s)

An atom in an Emerson-Lei condition, meaning 'the state s is visited infinitely often'.

U_i

A set of 'initialisation commands' in the reactive module corresponding to player i.

$\mathbb{Z}$

The integers: $\ldots, -1, 0, 1, \ldots$.

int(S)

The interior of the set S.

S

An iterated allocation structure.

mp(β)

The mean-payoff of an infinite sequence of real numbers, β; formally defined in Chapter 2, but can be thought of informally as a sort of infinite average.

$\models$

The modelling relation; used in various different forms, but in general, $\pi \models \alpha$ means that the path π adheres to the behaviour described by α.

$\mathbb{N}$

The natural numbers: $0, 1, \ldots$.

NE(G)

The set of Nash equilibria of a game G.

permset$\left(\rho\right)$

The set of states visited infinitely often in ρ.

Φ

In the setting of iterated Boolean games and reactive module games, the set of all propositional variables under consideration.

φ

A logical formula — the type of which depends on the context.

π

A path — a finite or infinite sequence of states; also used in the context of NTU games in the traditional definition of 'balancedness'.

$\pi(\vec{\sigma})$

The path induced by the players following the strategy profile $\vec{\sigma}$.

$\mathscr{P}(X)$

The power set of X; that is, the set containing all subsets of X.

pun_i

The best payoff player i can achieve, no matter what the remaining players do.

$[N]$

For $n \geq 1$, the set $\{1, 2, \ldots, n\}$.

req

A requirement function, detailing how many resources each atomic proposition needs to make it true in an iterated allocation structure.

$\vec{b}$

A resource bound.

R

A set of resources.

ρ

A run — an infinite sequence.

m_i

A reactive module corresponding to player i.

σ_i

A strategy for a player i.

s^0

The start state of a game.

St

A set of states of a game.

Q_i

The states of player i's finite memory strategy.

q_i^0

The initial state of player i's finite memory strategy.

δ_i

The transition function of player i's finite memory strategy.

tr

The transition function of a concurrent game structure — takes the current state of the game, along with an action for each agent/player, and outputs a new state for the game.

$\mathscr{T}$

A tree.

τ

A labelled tree.

$\top$

The logical formula, true.

U_i

A set of 'update commands' in the reactive module corresponding to player i.

val(α)

The valuation induced by an allocation α.

val(v)

The valuation induced by a vote v.

v_i

A vote for player i.

w_i

The weight function of player i, which defines that player's preferences.

$\mathscr{W}(v, s)$

The set of winning actions of a vote v in the state s.

Bibliography

[1] TSGM (https://web.archive.org/web/20220926221155/https://mathoverflow.net/users/31429/tsgm). *Why was John Nash's 1950 Game Theory paper such a big deal?* MathOverflow. (version: 2014-04-08). URL: https://web.archive.org/web/20220926221257/https://mathoverflow.net/questions/162836/why-was-john-nashs-1950-game-theory-paper-such-a-big-deal (cit. on p. 9).

[2] Alessandro Abate et al. 'Rational verification: game-theoretic verification of multi-agent systems'. In: *Appl. Intell.* 51.9 (2021), pp. 6569–6584. DOI: 10.1007/s10489-021-02658-y (cit. on p. 17).

[3] Milad Aghajohari, Guy Avni and Thomas A. Henzinger. 'Determinacy in Discrete-Bidding Infinite-Duration Games'. In: *Log. Methods Comput. Sci.* 17.1 (2021) (cit. on p. 171).

[4] Thomas Ågotnes and Natasha Alechina. 'Coalition logic with individual, distributed and common knowledge'. In: *J. Log. Comput.* 29.7 (2019), pp. 1041–1069. DOI: 10.1093/logcom/exv085 (cit. on p. 13).

[5] Thomas Ågotnes et al. 'Group announcement logic'. In: *J. Appl. Log.* 8.1 (2010), pp. 62–81. DOI: 10.1016/j.jal.2008.12.002 (cit. on p. 13).

[6] Natasha Alechina et al. 'Resource-Bounded Alternating-Time Temporal Logic'. In: *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1*. AAMAS '10. Toronto, Canada: International Found-

ation for Autonomous Agents and Multiagent Systems, 2010, pp. 481–488. ISBN: 9780982657119 (cit. on p. 23).

[7] Natasha Alechina et al. 'Logic for coalitions with bounded resources'. In: *Journal of Logic and Computation* 21.6 (2011), pp. 907–937. DOI: 10.1093/logcom/exq032 (cit. on pp. 13, 23).

[8] Natasha Alechina et al. 'On the complexity of resource-bounded logics'. In: *Theor. Comput. Sci.* 750 (2018), pp. 69–100. DOI: 10.1016/j.tcs.2018.01.019 (cit. on p. 23).

[9] Luca de Alfaro, Marco Faella and Mariëlle Stoelinga. 'Linear and Branching Metrics for Quantitative Transition Systems'. In: *Automata, Languages and Programming: 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings*. Ed. by Josep Díaz et al. Vol. 3142. Lecture Notes in Computer Science. Springer, 2004, pp. 97–109. DOI: 10.1007/978-3-540-27836-8_11 (cit. on p. 6).

[10] Luca de Alfaro and Thomas A. Henzinger. 'Concurrent Omega-Regular Games'. In: *15th Annual IEEE Symposium on Logic in Computer Science, Santa Barbara, California, USA, June 26-29, 2000*. IEEE Computer Society, 2000, pp. 141–154. DOI: 10.1109/LICS.2000.855763 (cit. on p. 12).

[11] Luca de Alfaro and Thomas A. Henzinger. 'Interface Theories for Component-Based Design'. In: *Embedded Software, First International Workshop, EMSOFT 2001, Tahoe City, CA, USA, October, 8-10, 2001, Proceedings*. Ed. by Thomas A. Henzinger and Christoph M. Kirsch. Vol. 2211. Lecture Notes in Computer Science. Springer, 2001, pp. 148–165. DOI: 10.1007/3-540-45449-7_11 (cit. on p. 173).

[12] Luca de Alfaro, Thomas A. Henzinger and Orna Kupferman. 'Concurrent reachability games'. In: *Theor. Comput. Sci.* 386.3 (2007), pp. 188–217. DOI: 10.1016/j.

tcs.2007.07.008. URL: https://doi.org/10.1016/j.tcs.2007.07.008 (cit. on p. 11).

[13] Luca de Alfaro and Rupak Majumdar. 'Quantitative solution of omega-regular games'. In: *J. Comput. Syst. Sci.* 68.2 (2004), pp. 374–397. DOI: 10.1016/j.jcss.2003.07.009. URL: https://doi.org/10.1016/j.jcss.2003.07.009 (cit. on p. 11).

[14] Luca de Alfaro et al. 'Model Checking Discounted Temporal Properties'. In: *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by Kurt Jensen and Andreas Podelski. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 77–92. ISBN: 978-3-540-24730-2 (cit. on p. 6).

[15] E. Allen Emerson and Chin-Laung Lei. 'Modalities for model checking: branching time logic strikes back'. In: *Science of Computer Programming* 8.3 (1987), pp. 275–306. ISSN: 0167-6423. DOI: https://doi.org/10.1016/0167-6423(87)90036-0 (cit. on pp. 8, 36).

[16] Shaull Almagor, Udi Boker and Orna Kupferman. 'Formalizing and Reasoning about Quality'. In: *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*. Ed. by Fedor V. Fomin et al. Vol. 7966. Lecture Notes in Computer Science. Springer, 2013, pp. 15–27. DOI: 10.1007/978-3-642-39212-2_3 (cit. on p. 24).

[17] Shaull Almagor, Orna Kupferman and Giuseppe Perelli. 'Synthesis of Controllable Nash Equilibria in Quantitative Objective Games'. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. IJCAI'18. Stockholm, Sweden: AAAI Press, 2018, pp. 35–41. ISBN: 9780999241127 (cit. on p. 24).

[18] Rajeev Alur, Tomás Feder and Thomas A. Henzinger. 'The Benefits of Relaxing Punctuality'. In: *J. ACM* 43.1 (1996), pp. 116–146. DOI: 10.1145/227595.227602 (cit. on p. 24).

[19] Rajeev Alur and Thomas A. Henzinger. 'Reactive Modules'. In: *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*. IEEE Computer Society, 1996, pp. 207–218. DOI: 10.1109/LICS.1996.561320 (cit. on pp. 14, 26, 96, 172, 173).

[20] Rajeev Alur, Thomas A. Henzinger and Orna Kupferman. 'Alternating-time temporal logic'. In: *J. ACM* 49.5 (2002), pp. 672–713. DOI: 10.1145/585265.585270 (cit. on pp. 12–14, 19, 20, 40, 121).

[21] Rajeev Alur and Salvatore La Torre. 'Deterministic generators and games for LTL fragments'. In: *ACM Transactions on Computational Logic (TOCL)* 5.1 (2004), pp. 1–25 (cit. on p. 19).

[22] Rajeev Alur et al. 'MOCHA: Modularity in Model Checking'. In: *Computer Aided Verification, 10th International Conference, CAV '98, Vancouver, BC, Canada, June 28 - July 2, 1998, Proceedings*. Ed. by Alan J. Hu and Moshe Y. Vardi. Vol. 1427. Lecture Notes in Computer Science. Springer, 1998, pp. 521–525. DOI: 10.1007/BFb0028774 (cit. on pp. 26, 173).

[23] Rajeev Alur et al. 'Syntax-Guided Synthesis'. In: *Dependable Software Systems Engineering*. Ed. by Maximilian Irlbeck, Doron A. Peled and Alexander Pretschner. Vol. 40. NATO Science for Peace and Security Series, D: Information and Communication Security. IOS Press, 2015, pp. 1–25. DOI: 10.3233/978-1-61499-495-4-1. URL: https://doi.org/10.3233/978-1-61499-495-4-1 (cit. on p. 174).

[24] Kenneth J. Arrow. 'An extension of the basic theorems of classical welfare economics'. In: *Proceedings of the second Berkeley symposium on mathematical statistics and probability*. Vol. 2. University of California Press. 1951, pp. 507–533 (cit. on p. 20).

[25] Kenneth J. Arrow and Gerard Debreu. 'Existence of an Equilibrium for a Competitive Economy'. In: *Econometrica* 22.3 (July 1954), p. 265 (cit. on p. 20).

[26] Robert J. Aumann. 'Acceptable points in general cooperative n-person games'. In: *Contributions to the theory of games, Vol. IV*. Annals of Mathematics Studies, no. 40. Princeton University Press, Princeton, N.J., 1959, pp. 287–324 (cit. on pp. 16, 20, 34).

[27] Robert J. Aumann. 'Acceptable points in games of perfect information'. In: *Pacific Journal of Mathematics* 10 (1960), pp. 381–417. ISSN: 0030-8730 (cit. on pp. 16, 20, 34).

[28] Guy Avni and Thomas A. Henzinger. 'A Survey of Bidding Games on Graphs (Invited Paper)'. In: *31st International Conference on Concurrency Theory, CONCUR 2020, September 1-4, 2020, Vienna, Austria (Virtual Conference)*. Ed. by Igor Konnov and Laura Kovács. Vol. 171. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 2:1–2:21. DOI: 10.4230/LIPIcs.CONCUR.2020.2 (cit. on pp. 25, 170).

[29] Guy Avni, Thomas A. Henzinger and Ventsislav Chonev. 'Infinite-duration Bidding Games'. In: *J. ACM* 66.4 (2019), 31:1–31:29. DOI: 10.1145/3340295 (cit. on pp. 25, 171).

[30] Guy Avni, Thomas A. Henzinger and Orna Kupferman. 'Dynamic resource allocation games'. In: *Theor. Comput. Sci.* 807 (2020), pp. 42–55. DOI: 10.1016/j.tcs.2019.06.031 (cit. on p. 25).

[31] Guy Avni, Thomas A. Henzinger and Đorđe Žikelić. 'Bidding mechanisms in graph games'. In: *Journal of Computer and System Sciences* 119 (Aug. 2021), pp. 133–144. DOI: 10.1016/j.jcss.2021.02.008 (cit. on p. 25).

[32] Robert Axelrod. *The Evolution Of Cooperation*. Basic Books, 1984. ISBN: 9780465021215 (cit. on p. 9).

[33] Tomás Babiak et al. 'The Hanoi Omega-Automata Format'. In: *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I*. Ed. by Daniel Kroening and Corina S. Pasareanu.

Vol. 9206. Lecture Notes in Computer Science. Springer, 2015, pp. 479–486. DOI: 10.1007/978-3-319-21690-4_31 (cit. on pp. 8, 36, 174).

[34] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008. ISBN: 978-0-262-02649-9 (cit. on pp. 6, 35).

[35] Haniel Barbosa et al. 'cvc5: A Versatile and Industrial-Strength SMT Solver'. In: *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I*. Ed. by Dana Fisman and Grigore Rosu. Vol. 13243. Lecture Notes in Computer Science. Springer, 2022, pp. 415–442. DOI: 10.1007/978-3-030-99524-9_24. URL: https://doi.org/10.1007/978-3-030-99524-9%5C_24 (cit. on p. 173).

[36] Clark Barrett, Aaron Stump and Cesare Tinelli. 'The SMT-LIB Standard: Version 2.0'. In: *Proceedings of the 8th International Workshop on Satisfiability Modulo Theories (Edinburgh, UK)*. Ed. by A. Gupta and Daniel Kroening. 2010 (cit. on p. 173).

[37] Clark Barrett et al. 'CVC4'. In: *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV '11)*. Ed. by Ganesh Gopalakrishnan and Shaz Qadeer. Vol. 6806. Lecture Notes in Computer Science. Snowbird, Utah. Springer, July 2011, pp. 171–177 (cit. on p. 3).

[38] Francesco Belardinelli et al. 'Strategy Logic with Simple Goals: Tractable Reasoning about Strategies'. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. Ed. by Sarit Kraus. ijcai.org, 2019, pp. 88–94. DOI: 10.24963/ijcai.2019/13. URL: https://doi.org/10.24963/ijcai.2019/13 (cit. on p. 14).

[39] Yves Bertot and Pierre Castran. *Interactive Theorem Proving and Program Development: Coq'Art The Calculus of Inductive Constructions*. 1st. Springer Publishing Company, Incorporated, 2010. ISBN: 3642058809 (cit. on p. 3).

[40] Louis J. Billera. 'Some theorems on the core of an n-game without side-payments'. In: *SIAM Journal on Applied Mathematics* 18.3 (1970). Publisher: SIAM, pp. 567–579 (cit. on pp. 91, 95, 96, 164).

[41] Roderick Bloem et al. 'Better Quality in Synthesis through Quantitative Objectives'. In: *Computer Aided Verification*. Ed. by Ahmed Bouajjani and Oded Maler. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 140–156. ISBN: 978-3-642-02658-4 (cit. on p. 133).

[42] Roderick Bloem et al. 'Synthesis of Reactive(1) designs'. In: *J. Comput. Syst. Sci.* 78.3 (2012), pp. 911–938. DOI: 10.1016/j.jcss.2011.08.007 (cit. on pp. 19, 36).

[43] Aaron Bohy et al. 'Acacia+, a Tool for LTL Synthesis'. In: *Computer Aided Verification - 24th International Conference, CAV 2012, Berkeley, CA, USA, July 7-13, 2012 Proceedings*. Ed. by P. Madhusudan and Sanjit A. Seshia. Vol. 7358. Lecture Notes in Computer Science. Springer, 2012, pp. 652–657. DOI: 10.1007/978-3-642-31424-7_45. URL: https://doi.org/10.1007/978-3-642-31424-7%5C_45 (cit. on p. 25).

[44] Aaron Bohy et al. 'Synthesis from LTL Specifications with Mean-Payoff Objectives'. In: *Tools and Algorithms for the Construction and Analysis of Systems - 19th International Conference, TACAS 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings*. Ed. by Nir Piterman and Scott A. Smolka. Vol. 7795. Lecture Notes in Computer Science. Springer, 2013, pp. 169–184. DOI: 10.1007/978-3-642-36742-7_12. URL: https://doi.org/10.1007/978-3-642-36742-7%5C_12 (cit. on pp. 24, 26).

[45] Udi Boker et al. 'Temporal Specifications with Accumulative Values'. In: *ACM Transactions on Computational Logic* 15.4 (Aug. 2014), pp. 1–25. DOI: 10.1145/2629686 (cit. on p. 23).

[46] O. N. Bondareva. 'Some applications of the methods of linear programming to the theory of cooperative games'. In: *Problemy Kibernet. No.* 10 (1963), pp. 119–139 (cit. on pp. 95, 164).

[47] Elise Bonzon et al. 'Boolean Games Revisited'. In: *ECAI 2006, 17th European Conference on Artificial Intelligence, August 29 - September 1, 2006, Riva del Garda, Italy, Including Prestigious Applications of Intelligent Systems (PAIS 2006), Proceedings*. Ed. by Gerhard Brewka et al. Vol. 141. Frontiers in Artificial Intelligence and Applications. IOS Press, 2006, pp. 265–269 (cit. on p. 140).

[48] Patricia Bouyer et al. 'Infinite Runs in Weighted Timed Automata with Energy Constraints'. In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 33–47. DOI: 10.1007/978-3-540-85778-5_4 (cit. on pp. 21, 22).

[49] Patricia Bouyer et al. 'Nash Equilibria in Concurrent Games with Büchi Objectives'. In: *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2011, December 12-14, 2011, Mumbai, India*. Ed. by Supratik Chakraborty and Amit Kumar. Vol. 13. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011, pp. 375–386. DOI: 10.4230/LIPIcs.FSTTCS.2011.375 (cit. on pp. 19, 26).

[50] Patricia Bouyer et al. 'Pure Nash Equilibria in Concurrent Deterministic Games'. In: *Log. Methods Comput. Sci.* 11.2 (2015). DOI: 10.2168/LMCS-11(2:9)2015 (cit. on pp. 9, 19, 26).

[51] Patricia Bouyer et al. 'Reachability in Networks of Register Protocols under Stochastic Schedulers'. In: *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*. Ed. by Ioannis Chatzigiannakis et al. Vol. 55. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, 106:1–106:14. DOI: 10.4230/LIPIcs.ICALP.2016.106 (cit. on p. 19).

[52] Romain Brenguier. 'PRALINE: A Tool for Computing Nash Equilibria in Concurrent Games'. In: *Computer Aided Verification*. Ed. by Natasha Sharygina and Helmut Veith. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 890–895. ISBN: 978-3-642-39799-8 (cit. on pp. 26, 173).

[53] Romain Brenguier and Jean-François Raskin. 'Pareto Curves of Multidimensional Mean-Payoff Games'. In: *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part II*. Ed. by Daniel Kroening and Corina S. Pasareanu. Vol. 9207. Lecture Notes in Computer Science. Springer, 2015, pp. 251–267. DOI: 10.1007/978-3-319-21668-3_15 (cit. on pp. 78, 90, 92, 164).

[54] Léonard Brice, Jean-François Raskin and Marie van den Bogaard. 'The Complexity of SPEs in Mean-Payoff Games'. In: *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*. Ed. by Mikolaj Bojanczyk, Emanuela Merelli and David P. Woodruff. Vol. 229. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 116:1–116:20. DOI: 10.4230/LIPIcs.ICALP.2022.116. URL: https://doi.org/10.4230/LIPIcs.ICALP.2022.116 (cit. on pp. 19, 22).

[55] Véronique Bruyère, Jean-François Raskin and Clément Tamines. 'Stackelberg-Pareto Synthesis'. In: *32nd International Conference on Concurrency Theory, CONCUR 2021, August 24-27, 2021, Virtual Conference*. Ed. by Serge Haddad and Daniele Varacca. Vol. 203. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 27:1–27:17. DOI: 10.4230/LIPIcs.CONCUR.2021.27 (cit. on p. 20).

[56] Véronique Bruyère, Jean-François Raskin and Clément Tamines. 'Pareto-Rational Verification'. In: *CoRR* abs/2202.13485 (2022). arXiv: 2202.13485 (cit. on pp. 17, 20).

[57] J. Richard Büchi. 'Decision methods in the theory of ordinals'. In: *Bull. Amer. Math. Soc.* 71.5 (Sept. 1965), pp. 767–770 (cit. on p. 9).

[58] J. Richard Büchi and Lawrence H. Landweber. 'Solving Sequential Conditions by Finite-State Strategies'. In: *The Collected Works of J. Richard Büchi*. Ed. by Saunders Mac Lane and Dirk Siefkes. New York, NY: Springer New York, 1990, pp. 525–541. ISBN: 978-1-4613-8928-6. DOI: 10.1007/978-1-4613-8928-6_29 (cit. on p. 9).

[59] Nils Bulling and Valentin Goranko. 'How to Be Both Rich and Happy: Combining Quantitative and Qualitative Strategic Reasoning about Multi-Player Games (Extended Abstract)'. In: *Proceedings 1st International Workshop on Strategic Reasoning, SR 2013, Rome, Italy, March 16-17, 2013*. Ed. by Fabio Mogavero, Aniello Murano and Moshe Y. Vardi. Vol. 112. EPTCS. 2013, pp. 33–41. DOI: 10.4204/EPTCS.112.8 (cit. on p. 23).

[60] Samuel R. Buss. 'The Boolean Formula Value Problem Is in ALOGTIME'. In: *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*. Ed. by Alfred V. Aho. ACM, 1987, pp. 123–131. DOI: 10.1145/28395.28409 (cit. on p. 40).

[61] Petr Čermák, Alessio Lomuscio and Aniello Murano. 'Verifying and Synthesising Multi-Agent Systems against One-Goal Strategy Logic Specifications'. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI'15. Austin, Texas: AAAI Press, 2015, pp. 2038–2044. ISBN: 0262511290 (cit. on p. 26).

[62] Arindam Chakrabarti et al. 'Resource Interfaces'. In: *Embedded Software*. Springer Berlin Heidelberg, 2003, pp. 117–133. DOI: 10.1007/978-3-540-45212-6_9 (cit. on p. 21).

[63] Toong Shoon Chan and Ian Gorton. 'Formal Validation of a High Performance Error Control Protocol Using SPIN'. In: *Software: Practice and Experience* 26.1 (1996), pp. 105–124. DOI: 10.1002/(SICI)1097-024X(199601)26:1<105::AID-SPE3>

3.0.CO;2-\#. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/\%28SICI\%291097-024X\%28199601\%2926\%3A1\%3C105\%3A\%3AAID-SPE3\%3E3.0.CO\%3B2-\%23 (cit. on p. 7).

[64] Krishnendu Chatterjee and Thomas A. Henzinger. 'A survey of stochastic ω-regular games'. In: *J. Comput. Syst. Sci.* 78.2 (2012), pp. 394–413. DOI: 10.1016/j.jcss.2011.05.002 (cit. on p. 11).

[65] Krishnendu Chatterjee, Thomas A. Henzinger and Marcin Jurdzinski. 'Mean-payoff parity games'. In: *20th Annual IEEE Symposium on Logic in Computer Science (LICS'05)*. IEEE, 2005, pp. 178–187 (cit. on pp. 11, 22, 133).

[66] Krishnendu Chatterjee, Thomas A. Henzinger and Nir Piterman. 'Algorithms for Büchi Games'. In: *CoRR* abs/0805.2620 (2008). arXiv: 0805.2620. URL: http://arxiv.org/abs/0805.2620 (cit. on p. 11).

[67] Krishnendu Chatterjee, Thomas A. Henzinger and Nir Piterman. 'Strategy logic'. In: *Inf. Comput.* Leibniz International Proceedings in Informatics (LIPIcs) 208.6 (2010). Ed. by Kamal Lodaya and Meena Mahajan, pp. 677–693. ISSN: 1868-8969. DOI: 10.1016/j.ic.2009.07.004 (cit. on pp. 14, 22).

[68] Krishnendu Chatterjee, Marcin Jurdzinski and Thomas A. Henzinger. 'Simple Stochastic Parity Games'. In: *Computer Science Logic, 17th International Workshop, CSL 2003, 12th Annual Conference of the EACSL, and 8th Kurt Gödel Colloquium, KGC 2003, Vienna, Austria, August 25-30, 2003, Proceedings*. Ed. by Matthias Baaz and Johann A. Makowsky. Vol. 2803. Lecture Notes in Computer Science. Springer, 2003, pp. 100–113. DOI: 10.1007/978-3-540-45220-1_11. URL: https://doi.org/10.1007/978-3-540-45220-1%5C_11 (cit. on p. 11).

[69] David Chaum. 'The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability'. In: *J. Cryptology* 1.1 (1988), pp. 65–75. DOI: 10.1007/BF00206326 (cit. on p. 26).

[70] Chris Chilton, Bengt Jonsson and Marta Z. Kwiatkowska. 'Assume-Guarantee Reasoning for Safe Component Behaviours'. In: *Formal Aspects of Component Software*. Springer Berlin Heidelberg, 2013, pp. 92–109. DOI: 10.1007/978-3-642-35861-6_6 (cit. on p. 173).

[71] Yunja Choi. 'From NuSMV to SPIN: Experiences with model checking flight guidance systems'. In: *Formal Methods in System Design* 30.3 (Jan. 2007), pp. 199–216. DOI: 10.1007/s10703-006-0027-9 (cit. on p. 7).

[72] Alonzo Church. 'Logic, arithmetic, and automata'. In: *Proceedings of the International Congress of Mathematicians, 15–22 August 1962, Institut Mittag-Leffler, Djursholm, Sweden*. Vol. 29. 1963, pp. 23–25 (cit. on p. 7).

[73] Alessandro Cimatti et al. 'NuSMV 2: An OpenSource Tool for Symbolic Model Checking'. In: *Computer Aided Verification, 14th International Conference, CAV 2002,Copenhagen, Denmark, July 27-31, 2002, Proceedings*. Ed. by Ed Brinksma and Kim Guldstrand Larsen. Vol. 2404. Lecture Notes in Computer Science. Springer, 2002, pp. 359–364. DOI: 10.1007/3-540-45657-0_29 (cit. on pp. 3, 6, 7).

[74] Edmund M. Clarke and E. Allen Emerson. 'Design and synthesis of synchronization skeletons using branching time temporal logic'. In: *Logics of Programs*. Springer-Verlag, 1981, pp. 52–71. DOI: 10.1007/bfb0025774 (cit. on pp. 5, 6, 8).

[75] Edmund M. Clarke, Daniel Kroening and Flavio Lerda. 'A Tool for Checking ANSI-C Programs'. In: *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2004)*. Ed. by Kurt Jensen and Andreas Podelski. Vol. 2988. Lecture Notes in Computer Science. Springer, 2004, pp. 168–176. ISBN: 3-540-21299-X (cit. on p. 3).

[76] Stephen A. Cook. 'The Complexity of Theorem-Proving Procedures'. In: *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker*

Heights, Ohio, USA. Ed. by Michael A. Harrison, Ranan B. Banerji and Jeffrey D. Ullman. ACM, 1971, pp. 151–158. DOI: 10.1145/800157.805047 (cit. on pp. v, 40, 58, 143).

[77] Katie Costello and Meghan Rimol. *Gartner Says Global IT Spending to Reach $3.9 Trillion in 2020*. 15th Jan. 2020 (cit. on p. 1).

[78] Richard Dawkins. *The Selfish Gene*. Oxford University Press, Oxford, UK, 1976 (cit. on p. 9).

[79] Gerard Debreu. 'The Coefficient of Resource Utilization'. In: Cowles Commission papers (1951) (cit. on p. 20).

[80] Alexandre Donzé. 'On Signal Temporal Logic'. In: *Runtime Verification*. Ed. by Axel Legay and Saddek Bensalem. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 382–383. ISBN: 978-3-642-40787-1 (cit. on p. 24).

[81] Rodney G. Downey and Michael R. Fellows. 'Fixed-Parameter Tractability and Completeness I: Basic Results'. In: *SIAM J. Comput.* 24.4 (1995), pp. 873–921. DOI: 10.1137/S0097539792228228 (cit. on p. 178).

[82] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer New York, 1999. DOI: 10.1007/978-1-4612-0515-9 (cit. on p. 19).

[83] Mark Dowson. 'The Ariane 5 software failure'. In: *ACM SIGSOFT Software Engineering Notes* 22.2 (1997), p. 84 (cit. on p. 2).

[84] Laurent Doyen et al. 'Realizability of Real-Time Logics'. In: *Formal Modeling and Analysis of Timed Systems*. Ed. by Joël Ouaknine and Frits W. Vaandrager. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 133–148. ISBN: 978-3-642-04368-0 (cit. on p. 24).

[85] Paul E. Dunne et al. 'Solving coalitional resource games'. In: *Artif. Intell.* 174.1 (2010), pp. 20–50. DOI: 10.1016/j.artint.2009.09.005 (cit. on p. 108).

[86] Rüdiger Ehlers. 'Unbeast: Symbolic Bounded Synthesis'. In: *Tools and Algorithms for the Construction and Analysis of Systems*. Springer Berlin Heidelberg, 2011, pp. 272–275. DOI: 10.1007/978-3-642-19835-9_25. URL: https://doi.org/10.1007/978-3-642-19835-9_25 (cit. on p. 25).

[87] Andrzej Ehrenfeucht and Jan Mycielski. 'Positional strategies for mean payoff games'. In: *International Journal of Game Theory* 8.2 (June 1979), pp. 109–113. DOI: 10.1007/bf01768705 (cit. on pp. 11, 21, 31, 48, 49, 60).

[88] E. Allen Emerson. 'Model Checking and the Mu-calculus'. In: *Descriptive Complexity and Finite Models, Proceedings of a DIMACS Workshop 1996, Princeton, New Jersey, USA, January 14-17, 1996*. Ed. by Neil Immerman and Phokion G. Kolaitis. Vol. 31. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. DIMACS/AMS, 1996, pp. 185–214. DOI: 10.1090/dimacs/031/06 (cit. on p. 6).

[89] E. Allen Emerson and Joseph Y. Halpern. '"Sometimes" and "not never" revisited'. In: *Journal of the ACM (JACM)* 33.1 (Jan. 1986), pp. 151–178. DOI: 10.1145/4904.4999 (cit. on p. 6).

[90] E. Allen Emerson and Charanjit S. Jutla. 'The Complexity of Tree Automata and Logics of Programs (Extended Abstract)'. In: *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*. IEEE Computer Society, 1988, pp. 328–337. DOI: 10.1109/SFCS.1988.21949 (cit. on p. 11).

[91] Ronald Fagin et al. *Reasoning About Knowledge*. MIT Press, 1995. ISBN: 9780262562003. DOI: 10.7551/mitpress/5803.001.0001 (cit. on p. 26).

[92] Emmanuel Filiot, Raffaella Gentilini and Jean-François Raskin. 'Rational Synthesis Under Imperfect Information'. In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*. Ed. by

Anuj Dawar and Erich Grädel. ACM, 2018, pp. 422–431. DOI: 10.1145/3209108.3209164. URL: https://doi.org/10.1145/3209108.3209164 (cit. on p. 20).

[93] Emmanuel Filiot, Naiyong Jin and Jean-François Raskin. 'An Antichain Algorithm for LTL Realizability'. In: *Computer Aided Verification*. Ed. by Ahmed Bouajjani and Oded Maler. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 263–277. ISBN: 978-3-642-02658-4 (cit. on p. 25).

[94] Bernd Finkbeiner and Sven Schewe. 'Bounded synthesis'. English. In: *International Journal on Software Tools for Technology Transfer* 15.5-6 (2013), pp. 519–539. ISSN: 1433-2779. DOI: 10.1007/s10009-012-0228-z. URL: http://dx.doi.org/10.1007/s10009-012-0228-z (cit. on p. 26).

[95] Dana Fisman, Orna Kupferman and Yoad Lustig. 'Rational Synthesis'. In: *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*. Ed. by Javier Esparza and Rupak Majumdar. Vol. 6015. Lecture Notes in Computer Science. Springer, 2010, pp. 190–204. DOI: 10.1007/978-3-642-12002-2_16 (cit. on pp. 5, 8, 17, 18, 43, 46, 121).

[96] Tong Gao, Julian Gutierrez and Michael J. Wooldridge. 'Iterated Boolean Games for Rational Verification'. In: *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*. Ed. by Kate Larson et al. ACM, 2017, pp. 705–713 (cit. on p. 17).

[97] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. ISBN: 0-7167-1044-7 (cit. on p. 57).

[98] Neil Ghani et al. 'Compositional Game Theory'. In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-*

12, 2018. Ed. by Anuj Dawar and Erich Grädel. ACM, 2018, pp. 472–481. DOI: 10.1145/3209108.3209165 (cit. on p. 173).

[99] Donald B. Gillies. '3. Solutions to General Non-Zero-Sum Games'. In: *Contributions to the Theory of Games (AM-40), Volume IV*. Ed. by Albert William Tucker and Robert Duncan Luce. Princeton University Press, Dec. 1959, pp. 47–86. DOI: 10.1515/9781400882168-005 (cit. on pp. 15, 20, 33).

[100] Valentin Goranko. 'Coalition games and alternating temporal logics'. In: *Proceedings of the 8th conference on Theoretical aspects of rationality and knowledge*. Morgan Kaufmann Publishers Inc. 2001, pp. 259–272 (cit. on p. 13).

[101] Julian Gutierrez, Paul Harrenstein and Michael J. Wooldridge. 'Iterated Boolean games'. In: *Inf. Comput.* 242 (2015), pp. 53–79. DOI: 10.1016/j.ic.2015.03.011 (cit. on pp. 17–20, 45, 46, 52, 108, 111, 116, 119).

[102] Julian Gutierrez, Paul Harrenstein and Michael J. Wooldridge. 'From model checking to equilibrium checking: Reactive modules for rational verification'. In: *Artif. Intell.* 248 (2017), pp. 123–157. DOI: 10.1016/j.artint.2017.04.003 (cit. on pp. 5, 17–19, 26, 43, 46, 48, 52, 96, 98).

[103] Julian Gutierrez, Sarit Kraus and Michael J. Wooldridge. 'Cooperative Concurrent Games'. In: *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*. Ed. by Edith Elkind et al. International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 1198–1206 (cit. on pp. 17, 20, 26, 33, 34, 45, 46, 52, 77, 86, 164).

[104] Julian Gutierrez, Giuseppe Perelli and Michael J. Wooldridge. 'Imperfect information in Reactive Modules games'. In: *Inf. Comput.* 261.Part (2018), pp. 650–675. DOI: 10.1016/j.ic.2018.02.023 (cit. on p. 21).

[105] Julian Gutierrez, Thomas Steeples and Michael J. Wooldridge. 'Mean-Payoff Games with ω-Regular Specifications'. In: *Games* 13.1 (2022), p. 19. DOI: 10.3390/g13010019 (cit. on pp. v, 165).

[106] Julian Gutierrez et al. 'Nash Equilibria in Concurrent Games with Lexicographic Preferences'. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. Publisher: Association for the Advancement of Artificial Intelligence. 2017, pp. 1067–1073. DOI: 10.24963/ijcai.2017/148 (cit. on pp. 22, 60).

[107] Julian Gutierrez et al. 'EVE: A Tool for Temporal Equilibrium Analysis'. In: *Automated Technology for Verification and Analysis - 16th International Symposium, ATVA 2018, Los Angeles, CA, USA, October 7-10, 2018, Proceedings*. Ed. by Shuvendu K. Lahiri and Chao Wang. Vol. 11138. Lecture Notes in Computer Science. Springer, 2018, pp. 551–557. DOI: 10.1007/978-3-030-01090-4_35 (cit. on pp. 26, 173).

[108] Julian Gutierrez et al. 'Equilibrium Design for Concurrent Games'. In: *30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands*. Ed. by Wan J. Fokkink and Rob van Glabbeek. Vol. 140. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 22:1–22:16. DOI: 10.4230/LIPIcs.CONCUR.2019.22 (cit. on pp. 34, 117, 120).

[109] Julian Gutierrez et al. 'Nash Equilibrium and Bisimulation Invariance'. In: *Log. Methods Comput. Sci.* 15.3 (2019). DOI: 10.23638/LMCS-15(3:32)2019 (cit. on pp. 18, 19).

[110] Julian Gutierrez et al. 'On Computational Tractability for Rational Verification'. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. Ed. by Sarit Kraus. ijcai.org, Aug. 2019, pp. 329–335. DOI: 10.24963/ijcai.2019/47 (cit. on pp. 17, 19, 22, 47, 61).

[111] Julian Gutierrez et al. *Automated Temporal Equilibrium Analysis: Verification and Synthesis of Multi-Player Games*. 2020. arXiv: 2008.05638 [cs.LO] (cit. on pp. 8, 26).

[112] Julian Gutierrez et al. 'Equilibria for games with combined qualitative and quantitative objectives'. In: *Acta Informatica* (June 2020). Publisher: Springer, pp. 1–26. DOI: 10.1007/s00236-020-00385-4 (cit. on pp. 22, 60, 133, 134).

[113] Julian Gutierrez et al. 'Rational Verification for Probabilistic Systems'. In: *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021*. Ed. by Meghyn Bienvenu, Gerhard Lakemeyer and Esra Erdem. 2021, pp. 312–322. DOI: 10.24963/kr.2021/30. URL: https://doi.org/10.24963/kr.2021/30 (cit. on pp. 17, 21).

[114] L. G. Hačijan. 'A polynomial algorithm in linear programming'. In: *Dokl. Akad. Nauk SSSR* 244.5 (1979), pp. 1093–1096. ISSN: 0002-3264 (cit. on p. 57).

[115] Hans Hansson and Bengt Jonsson. 'A Logic for Reasoning about Time and Reliability'. In: *Formal Asp. Comput.* 6.5 (1994), pp. 512–535. DOI: 10.1007/BF01211866 (cit. on p. 6).

[116] John Harrison. 'Formal Verification at Intel'. In: *18th IEEE Symposium on Logic in Computer Science (LICS 2003), 22-25 June 2003, Ottawa, Canada, Proceedings*. IEEE Computer Society, 2003, p. 45. DOI: 10.1109/LICS.2003.1210044 (cit. on p. 3).

[117] Thomas A. Henzinger. 'Games in system design and verification'. In: *Proceedings of the 10th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-2005), Singapore, June 10-12, 2005*. Ed. by Ron van der Meyden. National University of Singapore, 2005, pp. 1–4 (cit. on p. 12).

[118] Marijn J. H. Heule, Matti Järvisalo and Martin Suda. 'SAT Competition 2018'. In: *J. Satisf. Boolean Model. Comput.* 11.1 (2019), pp. 133–154. DOI: 10.3233/SAT190120. URL: https://doi.org/10.3233/SAT190120 (cit. on p. 174).

[119] Wiebe van der Hoek, Alessio Lomuscio and Michael J. Wooldridge. 'On the complexity of practical ATL model checking'. In: *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006*. Ed. by Hideyuki Nakashima et al. ACM, 2006, pp. 201–208. DOI: 10.1145/1160633.1160665 (cit. on pp. 96, 98).

[120] Gerard J. Holzmann. 'The Model Checker SPIN'. In: *IEEE Trans. Software Eng.* 23.5 (1997), pp. 279–295. DOI: 10.1109/32.588521 (cit. on pp. 3, 6, 7).

[121] Paul Hunter and Anuj Dawar. 'Complexity Bounds for Regular Games'. In: *Mathematical Foundations of Computer Science 2005*. Ed. by Joanna Jedrzejowicz and Andrzej Szepietowski. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 495–506. ISBN: 978-3-540-31867-5 (cit. on pp. 10, 150).

[122] Neil Immerman. 'Number of Quantifiers is Better Than Number of Tape Cells'. In: *J. Comput. Syst. Sci.* 22.3 (1981), pp. 384–406. DOI: 10.1016/0022-0000(81)90039-8. URL: https://doi.org/10.1016/0022-0000(81)90039-8 (cit. on p. 11).

[123] Matti Järvisalo et al. 'The International SAT Solver Competitions'. In: *AI Magazine* 33.1 (2012), pp. 89–92. DOI: 10.1609/aimag.v33i1.2395 (cit. on p. 3).

[124] Barbara Jobstmann and Roderick Bloem. 'Optimizations for LTL Synthesis'. In: *Formal Methods in Computer-Aided Design, 6th International Conference, FMCAD 2006, San Jose, California, USA, November 12-16, 2006, Proceedings*. IEEE Computer Society, 2006, pp. 117–124. DOI: 10.1109/FMCAD.2006.22 (cit. on p. 25).

[125] Marcin Jurdzinski. 'Deciding the Winner in Parity Games is in UP ∩co-UP'. In: *Inf. Process. Lett.* 68.3 (1998), pp. 119–124. DOI: 10.1016/S0020-0190(98)00150-1 (cit. on p. 11).

[126] Richard M. Karp. 'Reducibility Among Combinatorial Problems'. In: *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*. Ed. by

Raymond E. Miller and James W. Thatcher. The IBM Research Symposia Series. Plenum Press, New York, 1972, pp. 85–103. DOI: 10.1007/978-1-4684-2001-2_9 (cit. on p. 57).

[127] Richard M. Karp. 'A characterization of the minimum cycle mean in a digraph'. In: *Discrete mathematics* 23.3 (1978), pp. 309–311 (cit. on pp. 56, 60).

[128] Robert M. Keller. 'Formal Verification of Parallel Programs'. In: *Commun. ACM* 19.7 (July 1976), pp. 371–384. ISSN: 0001-0782. DOI: 10.1145/360248.360251 (cit. on pp. 5, 135).

[129] Yonit Kesten et al. 'A Decision Algorithm for Full Propositional Temporal Logic'. In: *Computer Aided Verification, 5th International Conference, CAV '93, Elounda, Greece, June 28 - July 1, 1993, Proceedings*. Ed. by Costas Courcoubetis. Vol. 697. Lecture Notes in Computer Science. Springer, 1993, pp. 97–109. DOI: 10.1007/3-540-56922-7_9 (cit. on p. 8).

[130] Phil Koopman. 'A case study of Toyota unintended acceleration and software safety'. In: (2014) (cit. on p. 2).

[131] Eryk Kopczynski. 'Half-Positional Determinacy of Infinite Games'. In: *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*. Ed. by Michele Bugliesi et al. Vol. 4052. Lecture Notes in Computer Science. Springer, 2006, pp. 336–347. DOI: 10.1007/11787006_29 (cit. on pp. 81, 90).

[132] Saul A. Kripke. 'Semantical Analysis of Modal Logic I Normal Modal Propositional Calculi'. In: *Mathematical Logic Quarterly* 9.5-6 (1963), pp. 67–96. DOI: 10.1002/malq.19630090502. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/malq.19630090502 (cit. on p. 5).

[133] Orna Kupferman, Giuseppe Perelli and Moshe Y. Vardi. 'Synthesis with rational environments'. In: *Ann. Math. Artif. Intell.* 78.1 (2016), pp. 3–20. DOI: 10.1007/s10472-016-9508-8 (cit. on p. 23).

[134] Orna Kupferman and Moshe Y. Vardi. 'Module Checking'. In: *Computer Aided Verification, 8th International Conference, CAV '96, New Brunswick, NJ, USA, July 31 - August 3, 1996, Proceedings*. Ed. by Rajeev Alur and Thomas A. Henzinger. Vol. 1102. Lecture Notes in Computer Science. Springer, 1996, pp. 75–86. DOI: 10.1007/3-540-61474-5_59 (cit. on p. 14).

[135] Orna Kupferman and Moshe Y. Vardi. 'Module Checking Revisited'. In: *Computer Aided Verification, 9th International Conference, CAV '97, Haifa, Israel, June 22-25, 1997, Proceedings*. Ed. by Orna Grumberg. Vol. 1254. Lecture Notes in Computer Science. Springer, 1997, pp. 36–47. DOI: 10.1007/3-540-63166-6_7 (cit. on p. 14).

[136] Orna Kupferman and Moshe Y. Vardi. 'Church's problem revisited'. In: *Bull. Symb. Log.* 5.2 (1999), pp. 245–263. DOI: 10.2307/421091. URL: https://doi.org/10.2307/421091 (cit. on pp. 7, 8).

[137] Orna Kupferman and Moshe Y. Vardi. 'Safraless Decision Procedures'. In: *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*. IEEE Computer Society, 2005, pp. 531–542. DOI: 10.1109/SFCS.2005.66. URL: https://doi.org/10.1109/SFCS.2005.66 (cit. on p. 26).

[138] Marta Z. Kwiatkowska et al. 'Automated Verification of Concurrent Stochastic Games'. In: *Quantitative Evaluation of Systems - 15th International Conference, QEST 2018, Beijing, China, September 4-7, 2018, Proceedings*. Ed. by Annabelle McIver and András Horváth. Vol. 11024. Lecture Notes in Computer Science. Springer, 2018, pp. 223–239. DOI: 10.1007/978-3-319-99154-2_14 (cit. on pp. 21, 27).

[139] Marta Z. Kwiatkowska et al. 'Equilibria-based Probabilistic Model Checking for Concurrent Stochastic Games'. In: *CoRR* abs/1811.07145 (2018). arXiv: 1811.07145 (cit. on pp. 21, 27).

[140] Marta Z. Kwiatkowska et al. 'PRISM-games 3.0: Stochastic Game Verification with Concurrency, Equilibria and Time'. In: *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part II*. Ed. by Shuvendu K. Lahiri and Chao Wang. Vol. 12225. Lecture Notes in Computer Science. Springer, 2020, pp. 475–487. DOI: 10.1007/978-3-030-53291-8_25 (cit. on p. 27).

[141] Lawrence H. Landweber. 'A design algorithm for sequential machines and definability in monadic second-order arithmetic'. In: 1967 (cit. on p. 9).

[142] Andrew J. Lazarus et al. 'Richman games'. In: *arXiv preprint math/9502222* (1995) (cit. on p. 25).

[143] Andrew J. Lazarus et al. 'Combinatorial games under auction play'. In: *Games and Economic Behavior* 27.2 (1999), pp. 229–264 (cit. on p. 25).

[144] Nancy G. Leveson and Clark Savage Turner. 'Investigation of the Therac-25 Accidents'. In: *Computer* 26.7 (1993), pp. 18–41. DOI: 10.1109/MC.1993.274940 (cit. on p. 1).

[145] Leonid A. Levin. 'Universal problems of full search'. Russian. In: *Probl. Peredachi Inf.* 9.3 (1973), pp. 115–116. ISSN: 0555-2923 (cit. on p. 143).

[146] Orna Lichtenstein and Amir Pnueli. 'Checking That Finite State Concurrent Programs Satisfy Their Linear Specification'. In: *Proceedings of the 12th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*. POPL '85. New Orleans, Louisiana, USA: Association for Computing Machinery, 1985, pp. 97–107. ISBN: 0897911474. DOI: 10.1145/318593.318622 (cit. on pp. 6, 40, 45).

[147] Alessio Lomuscio, Hongyang Qu and Franco Raimondi. 'MCMAS: A Model Checker for the Verification of Multi-Agent Systems'. In: *Computer Aided Verification*. Ed. by Ahmed Bouajjani and Oded Maler. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 682–688. ISBN: 978-3-642-02658-4 (cit. on p. 26).

[148] Alessio Lomuscio and Marek J. Sergot. 'Deontic Interpreted Systems'. In: *Studia Logica* 75.1 (2003), pp. 63–92. DOI: 10.1023/A:1026176900459 (cit. on p. 26).

[149] Oded Maler and Dejan Nickovic. 'Monitoring Temporal Properties of Continuous Signals'. In: *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, Joint International Conferences on Formal Modelling and Analysis of Timed Systems, FORMATS 2004 and Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT 2004, Grenoble, France, September 22-24, 2004, Proceedings*. Ed. by Yassine Lakhnech and Sergio Yovine. Vol. 3253. Lecture Notes in Computer Science. Springer, 2004, pp. 152–166. DOI: 10.1007/978-3-540-30206-3_12 (cit. on p. 24).

[150] Donald A. Martin. 'Borel Determinacy'. In: *Annals of Mathematics* 102.2 (1975), pp. 363–371. ISSN: 0003486X (cit. on p. 11).

[151] John Maynard Smith. *Evolution and the Theory of Games*. Cambridge University Press, 1982. ISBN: 978-0-51180629-2. DOI: 10.1017/CBO9780511806292 (cit. on p. 9).

[152] Robert McNaughton. 'Infinite games played on finite graphs'. In: *Annals of Pure and Applied Logic* 65.2 (1993). Publisher: Elsevier, pp. 149–184 (cit. on pp. 9–11, 40, 150, 162, 168).

[153] Nimrod Megiddo and Christos H. Papadimitriou. 'On total functions, existence theorems and computational complexity'. In: *Theoretical Computer Science* 81.2 (1991), pp. 317–324. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(91)90200-L (cit. on p. 49).

[154] Fabio Mogavero, Aniello Murano and Moshe Y. Vardi. 'Reasoning About Strategies'. In: *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*. Ed. by Kamal Lodaya and Meena Mahajan. Vol. 8. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010, pp. 133–144. DOI: 10.4230/LIPIcs.FSTTCS.2010.133 (cit. on pp. 14, 19).

[155] Fabio Mogavero et al. 'What Makes ATL* Decidable? A Decidable Fragment of Strategy Logic'. In: *CONCUR 2012 - Concurrency Theory - 23rd International Conference, CONCUR 2012, Newcastle upon Tyne, UK, September 4-7, 2012. Proceedings*. Ed. by Maciej Koutny and Irek Ulidowski. Vol. 7454. Lecture Notes in Computer Science. Springer, 2012, pp. 193–208. DOI: 10.1007/978-3-642-32940-1_15. URL: https://doi.org/10.1007/978-3-642-32940-1%5C_15 (cit. on p. 14).

[156] Fabio Mogavero et al. 'Reasoning About Strategies: On the Model-Checking Problem'. In: *ACM Trans. Comput. Log.* 15.4 (2014), 34:1–34:47. DOI: 10.1145/2631917 (cit. on pp. 8, 14, 19).

[157] Dov Monderer and Lloyd S. Shapley. 'Potential Games'. In: *Games and Economic Behavior* 14.1 (May 1996), pp. 124–143. DOI: 10.1006/game.1996.0044 (cit. on p. 25).

[158] Edward F. Moore. 'Gedanken-experiments on sequential machines'. In: *Automata studies*. Annals of Mathematics Studies, no. 34. Princeton University Press, Princeton, N.J., 1956, pp. 129–153 (cit. on p. 42).

[159] Leonardo Mendonça de Moura and Nikolaj S. Bjørner. 'Z3: An Efficient SMT Solver'. In: *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*. Ed. by C. R. Ramakrishnan and Jakob Rehof. Vol. 4963. Lecture

Notes in Computer Science. Springer, 2008, pp. 337–340. DOI: 10.1007/978-3-540-78800-3_24. URL: https://doi.org/10.1007/978-3-540-78800-3%5C_24 (cit. on pp. 3, 173).

[160] Leonardo Mendonça de Moura et al. 'The Lean Theorem Prover (System Description)'. In: *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*. Ed. by Amy P. Felty and Aart Middeldorp. Vol. 9195. Lecture Notes in Computer Science. Springer, 2015, pp. 378–388. DOI: 10.1007/978-3-319-21401-6_26 (cit. on p. 3).

[161] John F. Nash. 'Equilibrium points in n-person games'. In: *Proceedings of the national academy of sciences* 36.1 (1950), pp. 48–49. ISSN: 0027-8424. DOI: 10.1073/pnas.36.1.48. eprint: https://www.pnas.org/content/36/1/48.full.pdf (cit. on pp. 8, 15, 33).

[162] John F. Nash. 'Non-cooperative games'. In: *Annals of mathematics* (1951). Publisher: JSTOR, pp. 286–295 (cit. on pp. 8, 15, 33).

[163] Pavel Naumov and Jia Tao. 'Together we know how to achieve: An epistemic logic of know-how'. In: *Artif. Intell.* 262 (2018), pp. 279–300. DOI: 10.1016/j.artint.2018.06.007 (cit. on p. 13).

[164] John von Neumann. 'Zur Theorie der Gesellschaftsspiele'. In: 100 (1928), pp. 295–320. ISSN: 0025-5831. DOI: 10.1007/bf01448847 (cit. on p. 8).

[165] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior (60th-Anniversary Edition)*. Princeton University Press, 2007. ISBN: 978-0-691-13061-3 (cit. on pp. 8, 16).

[166] Frits de Nijs et al. 'Bounding the Probability of Resource Constraint Violations in Multi-Agent MDPs'. In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI'17. San Francisco, California, USA: AAAI Press, 2017, pp. 3562–3568 (cit. on p. 107).

[167] Frits de Nijs et al. 'Constrained Multiagent Markov Decision Processes: a Taxonomy of Problems and Algorithms'. In: *Journal of Artificial Intelligence Research* 70 (Mar. 2021), pp. 955–1001. DOI: 10.1613/jair.1.12233 (cit. on p. 107).

[168] Tobias Nipkow, Markus Wenzel and Lawrence C. Paulson. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Berlin, Heidelberg: Springer-Verlag, 2002. ISBN: 3540433767 (cit. on p. 3).

[169] Martin J. Osborne and Ariel Rubinstein. *A course in game theory*. MIT press, 1994 (cit. on pp. 20, 31, 86).

[170] Christos H. Papadimitriou and Mihalis Yannakakis. 'On the Approximability of Trade-offs and Optimal Access of Web Sources'. In: *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*. IEEE Computer Society, 2000, pp. 86–92. DOI: 10.1109/SFCS.2000.892068 (cit. on p. 20).

[171] Vilfredo Pareto. *Cours d'Economie politique, Volume I*. Duncker & Humblot, 1896 (cit. on p. 20).

[172] Vilfredo Pareto. *Cours d'Economie politique, Volume II*. Pichon, F., 1897 (cit. on p. 20).

[173] Marc Pauly. 'A Modal Logic for Coalitional Power in Games'. In: *J. Log. Comput.* 12.1 (2002), pp. 149–166. DOI: 10.1093/logcom/12.1.149 (cit. on p. 13).

[174] Charles Pecheur and Franco Raimondi. 'Symbolic Model Checking of Logics with Actions'. In: *Model Checking and Artificial Intelligence*. Ed. by Stefan Edelkamp and Alessio Lomuscio. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 113–128. ISBN: 978-3-540-74128-2 (cit. on p. 6).

[175] Dominique Perrin and Jean-Eric Pin. *Infinite words - automata, semigroups, logic and games*. Vol. 141. Pure and applied mathematics series. Elsevier Morgan Kaufmann, 2004. ISBN: 978-0-12-532111-2 (cit. on pp. 8, 9).

[176] Nir Piterman. 'From Nondeterministic Büchi and Streett Automata to Deterministic Parity Automata'. In: *Log. Methods Comput. Sci.* 3.3 (2007). DOI: 10.2168/LMCS-3(3:5)2007 (cit. on p. 22).

[177] Amir Pnueli. 'The Temporal Logic of Programs'. In: *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*. IEEE Computer Society, 1977, pp. 46–57. DOI: 10.1109/SFCS.1977.32 (cit. on pp. 5, 35).

[178] Amir Pnueli and Roni Rosner. 'On the Synthesis of a Reactive Module'. In: *Conference Record of the Sixteenth Annual ACM Symposium on Principles of Programming Languages, Austin, Texas, USA, January 11-13, 1989*. ACM Press, 1989, pp. 179–190. DOI: 10.1145/75277.75293 (cit. on pp. 8, 26, 40, 45, 121).

[179] Kevin Poulsen. *Software Bug Contributed to Blackout*. Feb. 2004 (cit. on p. 2).

[180] Arkadi Predtetchinski and P. Jean-Jacques Herings. 'A necessary and sufficient condition for non-emptiness of the core of a non-transferable utility game'. In: *J. Econ. Theory* 116.1 (2004), pp. 84–92. DOI: 10.1016/S0022-0531(03)00261-8 (cit. on p. 164).

[181] A. Rapoport. *Two-person Game Theory: The Essential Ideas*. Ann Arbor Paperbacks Series. University of Michigan Press, 1966. ISBN: 9780472050154 (cit. on p. 20).

[182] H. G. Rice. 'Classes of Recursively Enumerable Sets and Their Decision Problems'. In: *Transactions of the American Mathematical Society* 74.2 (1953), pp. 358–366. ISSN: 00029947 (cit. on p. 4).

[183] Robert W. Rosenthal. 'A class of games possessing pure-strategy Nash equilibria'. In: *International Journal of Game Theory* 2.1 (Dec. 1973), pp. 65–67. DOI: 10.1007/bf01737559 (cit. on p. 25).

[184] Roni Rosner. 'Modular synthesis of reactive systems'. In: (Jan. 1991) (cit. on pp. 8, 40, 45, 121).

[185] Lawrence M. Ruane. 'Process Synchronization in the UTS Kernel'. In: *Comput. Syst.* 3.3 (1990), pp. 387–421 (cit. on p. 7).

[186] Theo C. Ruys and Rom Langerak. 'Validation of Bosch' Mobile Communication Network Architecture with Spin'. In: *In Proceedings of SPIN97, the Third International Workshop on SPIN, University of Twente*. 1997 (cit. on p. 7).

[187] Shmuel Safra. 'On the Complexity of omega-Automata'. In: *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*. IEEE Computer Society, 1988, pp. 319–327. DOI: 10.1109/SFCS.1988.21948. URL: https://doi.org/10.1109/SFCS.1988.21948 (cit. on p. 26).

[188] Walter J. Savitch. 'Relationships between nondeterministic and deterministic tape complexities'. In: *Journal of Computer and System Sciences* 4.2 (Apr. 1970), pp. 177–192. DOI: 10.1016/s0022-0000(70)80006-x (cit. on p. 118).

[189] Herbert E. Scarf. 'The core of an N person game'. In: *Econometrica: Journal of the Econometric Society* (1967). Publisher: JSTOR, pp. 50–69 (cit. on pp. 91, 95, 164).

[190] Lloyd S. Shapley. 'On balanced sets and cores'. In: *Naval Research Logistics Quarterly* 14.4 (1967). Publisher: Wiley, pp. 453–460. DOI: 10.1002/nav.3800140404 (cit. on pp. 95, 164).

[191] Lloyd S. Shapley. 'On Balanced Games without Side Payments'. In: *Mathematical Programming*. Elsevier, 1973, pp. 261–290. DOI: 10.1016/b978-0-12-358350-5.50012-9 (cit. on pp. 91, 95, 164).

[192] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems - Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009. ISBN: 978-0-521-89943-7 (cit. on p. 12).

[193] A. Prasad Sistla and Edmund M. Clarke. 'The Complexity of Propositional Linear Temporal Logics'. In: *J. ACM* 32.3 (1985), pp. 733–749. DOI: 10.1145/3828.3837 (cit. on pp. 6, 40, 45, 83).

[194] Konrad Slind and Michael Norrish. 'A Brief Overview of HOL4'. In: *Theorem Proving in Higher Order Logics*. Ed. by Otmane Ait Mohamed, César Muñoz and Sofiène Tahar. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 28–32. ISBN: 978-3-540-71067-7 (cit. on p. 3).

[195] Jean Souyris et al. 'Formal Verification of Avionics Software Products'. In: *FM 2009: Formal Methods*. Ed. by Ana Cavalcanti and Dennis R. Dams. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 532–546. ISBN: 978-3-642-05089-3 (cit. on p. 3).

[196] Thomas Steeples, Julian Gutierrez and Michael J. Wooldridge. 'Mean-Payoff Games with ω-Regular Specifications'. In: *AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021*. Ed. by Frank Dignum et al. ACM, 2021, pp. 1272–1280. DOI: 10.5555/3463952.3464099 (cit. on pp. v, 165).

[197] Larry J. Stockmeyer and Ashok K. Chandra. 'Provably Difficult Combinatorial Games'. In: *SIAM J. Comput.* 8.2 (1979), pp. 151–174. DOI: 10.1137/0208013 (cit. on p. 103).

[198] Jan Strejcek. 'Linear temporal logic: Expressiveness and model checking'. PhD thesis. PhD thesis, Faculty of Informatics, Masaryk University in Brno, 2004 (cit. on p. 19).

[199] Synopsys Software Integrity Group. *The Heartbleed Bug*. 2014 (cit. on p. 2).

[200] Robert Endre Tarjan. 'Depth-First Search and Linear Graph Algorithms'. In: *SIAM J. Comput.* 1.2 (1972), pp. 146–160. DOI: 10.1137/0201010 (cit. on pp. 143, 158).

[201] Alan M. Turing. 'On Computable Numbers, with an Application to the Entscheidungsproblem'. In: *Proceedings of the London Mathematical Society* s2-42.1 (Jan. 1937), pp. 230–265. ISSN: 0024-6115. DOI: 10.1112/plms/s2-42.1.230. eprint: https:

//academic.oup.com/plms/article-pdf/s2-42/1/230/4317544/s2-42-1-230.pdf (cit. on p. 4).

[202] Michael Ummels and Dominik Wojtczak. 'The Complexity of Nash Equilibria in Limit-Average Games'. In: *CoRR*. Lecture Notes in Computer Science abs/1109.6220 (2011). Ed. by Joost-Pieter Katoen Katoen and Barbara König, pp. 482–496. DOI: 10.1007/978-3-642-23217-6_32 (cit. on pp. 9, 18, 22, 31, 47, 49, 55, 57, 60, 61, 67, 74).

[203] Moshe Y. Vardi. 'Alternating Automata and Program Verification'. In: *Computer Science Today: Recent Trends and Developments*. Ed. by Jan van Leeuwen. Vol. 1000. Lecture Notes in Computer Science. Springer, 1995, pp. 471–485. DOI: 10.1007/BFb0015261 (cit. on pp. 8, 118).

[204] Moshe Y. Vardi. 'Branching vs. Linear Time: Final Showdown'. In: *Tools and Algorithms for the Construction and Analysis of Systems, 7th International Conference, TACAS 2001 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2001 Genova, Italy, April 2-6, 2001, Proceedings*. Ed. by Tiziana Margaria and Wang Yi. Vol. 2031. Lecture Notes in Computer Science. Springer, 2001, pp. 1–22. DOI: 10.1007/3-540-45319-9_1 (cit. on p. 6).

[205] Moshe Y. Vardi and Pierre Wolper. 'An Automata-Theoretic Approach to Automatic Program Verification (Preliminary Report)'. In: *Proceedings of the Symposium on Logic in Computer Science (LICS '86), Cambridge, Massachusetts, USA, June 16-18, 1986*. IEEE Computer Society, 1986, pp. 332–344 (cit. on p. 8).

[206] Moshe Y. Vardi and Pierre Wolper. 'Reasoning About Infinite Computations'. In: *Inf. Comput.* 115.1 (1994), pp. 1–37. DOI: 10.1006/inco.1994.1092 (cit. on p. 8).

[207] Yaron Velner et al. 'The complexity of multi-mean-payoff and multi-energy games'. In: *Inf. Comput.* 241 (2015), pp. 177–196. DOI: 10.1016/j.ic.2015.03.001 (cit. on pp. 11, 22, 31, 80, 81, 90, 164).

[208] Tjark Weber et al. 'The SMT Competition 2015-2018'. In: *J. Satisf. Boolean Model. Comput.* 11.1 (2019), pp. 221–259. URL: https://doi.org/10.3233/SAT190123 (cit. on p. 174).

[209] Michael J. Wooldridge. *An Introduction to MultiAgent Systems, Second Edition*. Wiley, 2009. ISBN: 978-0-470-51946-2 (cit. on p. 12).

[210] Michael J. Wooldridge and Paul E. Dunne. 'On the computational complexity of coalitional resource games'. In: *Artif. Intell.* 170.10 (2006), pp. 835–871. DOI: 10.1016/j.artint.2006.03.003 (cit. on p. 108).

[211] Michael J. Wooldridge et al. 'Rational Verification: From Model Checking to Equilibrium Checking'. In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*. Ed. by Dale Schuurmans and Michael P. Wellman. AAAI Press, 2016, pp. 4184–4191 (cit. on pp. 17, 18).

[212] Wieslaw Zielonka. 'Infinite Games on Finitely Coloured Graphs with Applications to Automata on Infinite Trees'. In: *Theor. Comput. Sci.* 200.1-2 (1998), pp. 135–183. DOI: 10.1016/S0304-3975(98)00009-7 (cit. on p. 11).

[213] Uri Zwick and Mike Paterson. 'The complexity of mean payoff games on graphs'. In: *Theoretical Computer Science* 158.1-2 (May 1996), pp. 343–359. DOI: 10.1016/0304-3975(95)00188-3 (cit. on pp. 11, 21, 31, 48, 49).