PIOTR KAŹMIERCZAK

AGENTS THAT PLAY BY THE RULES

Dedicated to the loving memory of Leon Tadeusz Pawlak.
1918 – 2006

SCIENTIFIC ENVIRONMENT

The research presented in this dissertation has been conducted at the Department of Computing, Mathematics and Physics, Bergen University College under the research program Software Technologies for Distributed Systems (DISTECH) in cooperation with Research group for Logic, Language and Information (LII), Department of Information Science and Media Studies, University of Bergen.

This research was supported by the Research Council of Norway project number 194521 (FORMGRID), and by the Meltzer Research Fund.

ABSTRACT

This thesis is about coordination of multi-agent systems in general, and about the *social laws* paradigm for coordination in particular.

We present a logical and game-theoretic results about conceptual, technical and practical aspects of coordination. This thesis is paper-based, and consists of three introductory chapters and four papers which together provide the necessary background, motivations, and results.

We analyze philosophical underpinnings of coordination and cooperation in the popular strategic logic ATL, show that multi-agency requires limited coordination, and back this claim formally by presenting a truth-preserving translation from a variant of ATL into a language (and corresponding class of models) which supports only singleton coalitions. We then move to the analysis of social laws and introduce a novel approach to laws which take other agents' actions into account. We show how this new concept can be applied and how game-theoretic tools can be used to ensure efficient algorithmic solutions to related decision problems. Furthermore, we study a class of cooperative games induced by social laws, analyze their properties and computational problems. We emphasize the advantages of game-theoretic aspects of social laws as well as point out potential issues that arise while using game-theoretic concepts. Finally, we present a model checker implemented in Haskell, argue for this particular language's suitability and point out strengths of our tool.

The thesis is structured as follows. In Chapter 1 we introduce all the necessary background material related to multi-agent systems, the usage of formal logic and game theory to reason about them as well as motivations for our work. In Chapter 2 we present our results, and conclude in Chapter 3. Four papers follow.

KEYWORDS: alternating-time temporal logic, anonymous games, computational complexity, concurrent game structures, cooperative games, coordination, formal logic, functional programming, game theory, Haskell, model checking, multi-agent systems, norm compliance, normal form games, social laws, strategic logics, tractability

PUBLICATIONS

This thesis is based on the following papers:

- Piotr Kaźmierczak, Thomas Ågotnes and Wojciech Jamroga: "Multi-Agency Is Coordination And (Limited) Communication", to appear in the Proceedings of the 17th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA 2014), December 1–5, 2014, Gold Coast, Queensland, Australia.
- Sjur Dyrkolbotn and Piotr Kaźmierczak: "Playing with norms: Tractability of normative systems for homogeneous game structures", in Proceedings of the 13th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2014), May 5–9, 2014, Paris, France.
- Piotr Kaźmierczak: "Compliance Games", submitted to the 12th European Conference on Multi-Agent Systems (EUMAS 2014), December 18–19, 2014, Prague, Czech Republic.
- Piotr Kaźmierczak, Truls Pedersen and Thomas Ågotnes: "NorMC: a Norm Compliance Temporal Logic Model Chcker", in Proceedings of the Sixth Starting Artificial Intelligence Research Symposium (STAIRS 2012), August 27–28, 2012, Montpellier, France.

ACKNOWLEDGMENTS

Writing a Ph.D. thesis is no easy task, and, as is the case for many students, numerous people contributed, either directly or indirectly, to me completing this work.

This thesis would not have been written if it was not for my supervisor, Thomas Ågotnes. Thank you for all the discussions, comments, and feedback, but most importantly for loads of indulgence and patience. Thank you also for being a good friend, and for motivating me to work when I felt down and uninspired (roughly every 6 months or so).

Secondly, I would like to thank everyone at Høgskolen i Bergen's department of Computing, Mathematics and Physics, especially to the head of department, Carsten Helgesen, for being a great and very helpful boss. To Lars Michael Kristensen, head of our DISTECH project, and Yngve Lamo, leader of the FORMGRID project, I extend special thanks for providing financial and administrative support. Speaking of practical matters, I cannot forget about Elin Fauskanger Thane, head of administration in our department—I apologize for all the incorrectly filled forms and thank you for helping out with any and all paperwork.

Thank you to the people I taught with: Hege Erdal, Håvard Helstrup, Bjarte Kileng, Lars Michael Kristensen, Yngve Lamo, Adrian Rutle, Kent Inge Simonsen, Jon Eivind Vatne, and Xiaoliang Wang. I learned a lot from working with you and you made my teaching experience a pleasure.

Finally thank yous go to friends and co-authors. The Bergen Logic Group—Pål Grønås Drange, Sjur Dyrkolbotn, Ragnhild Iveranna Hogstad Jordahl, Erik Parmann, Truls Pedersen, Andrew Polonsky, Marija Slavkovik, Lubos Steskal, Paul Simon Svanberg, Yì Wáng and Zuojun Xiong—thanks for all the seminars and meetings, discussion and laughs. Special thanks to my "unofficial co-supervisors", Sjur & Truls, whom I have co-authored papers with and discussed almost every topic I worked on. Huge thanks to Wojtek Jamroga for fruitful collaboration and motivating discussions.

Thank you my dear office colleagues at HiB—Erik Eikeland, Fazle Rabbi, Hege Erdal, Ajith Kumar, Florian Mantz, Adrian Rutle, Kent Inge Simonsen, Sami Taktak, and Xiaoliang Wang—for all the lunches and office hours we shared. Thank you Hege Erdal and Jan Christian Liby for all the board game nights. Thank you Hannah Amanda Hansen for many discussions about cognitive science or research in general, and sending tons of funny cat pictures. Thank you Puja Gupta for the insane amounts of great Indian food and for showing me how research is done at the molecular biology department, and thank you Samia Touileb for all the awesome cakes. Thank you Pim van 't Hof for discussions about classical music and Rotterdam. Thank you Maja Jaakson for your Canadian accent. Thank you Lars Michael Kristensen for all the bike rides we had together

and for convincing me to try the Bergen-Voss race. Thank you Michał Pilipczuk for all the hikes.

Last but not least, I wish to extend special gratitude to the people closest to me. To my parents Alicja & Bernard for not asking about progress in my thesis too often, and to my fiancée Karolina Krzyżanowska for constant support, many fruitful discussions about research, philosophy, politics and music, and for all the time shared together despite living in two different countries working on our respective Ph.D. theses.

Thank you once more. You have all been very helpful during these four years (i. e., also indirectly responsible for the contents of this thesis), and I am pretty sure I would not have made it without you.

Piotr Kaźmierczak

CONTENTS

Part I

OVERVIEW

1

INTRODUCTION: MOTIVATIONS AND BACKGROUND

This chapter starts with three sections that cover background material on multi-agent systems, logical frameworks for multi-agent systems, and basic game-theoretic concepts. We describe what multi-agent systems are, explain why research in the area is important and explain how logical and game-theoretic methods can be applied to reason about agents. The chapter ends with a presentation of the state-of-the-art in combining logic and game theory to reason about social laws—the paradigm which is studied in this thesis—and introduces the reader to all the relevant conceptual and formal material necessary to understand the presentation of results covered in Chapter 2.

1.1 MULTI-AGENT SYSTEMS

We start our introduction to multi-agent systems by writing the simplest possible definition: multi-agent systems are systems comprised of more than one agent. This naturally begs the question: what are *agents?* Before we go into formal definitions, we provide an informal description and an example, to provide some intuitions.

Why do we need multi-agent systems?

In broadest terms (intelligent) agents are computer systems that can make their own decisions and pursue goals. As much as we like computers to be obedient and follow algorithms they are programmed with to the letter, there is a growing number of situations in which we would like our software to adapt its behavior to the environment. Picture an autonomous controller of the Mars Rover landing module. If anything out of the ordinary happens during its mission, we do not want the software to crash or quit. We would rather prefer it devised a strategy to pursue its initial goal in a different way than originally programmed, or to inform the Earth control center that the initial goal is not achievable, and that a new one must be given. This simplistic example illustrates what is perhaps the biggest challenge in designing intelligent agents: achieving the balance between agents' ability to pursue goal and their *reactiveness*, i. e., ability to adapt to new circumstances. We will return to this problem in a moment.

Formally, agents can be defined in the following way (Wooldridge & Jennings [111]):

> An *agent* is a computer system that is situated in some *environment* and that is capable of *autonomous action* in this environment in order to achieve its delegated objectives.

We emphasized a couple of notions in the above definition that need to be carefully explained.

Agents vs. intelligent agents

Firstly, the definition refers to *agents*, and not necessarily *intelligent agents* as in our small example above. Non-intelligent agents are an old idea in computer science. Such agents are pieces of software that simply react to certain programmed stimuli. A good and often used example of a simple software agent can be a program that controls the thermostat—its actions are to either turn the heating up, down, off or on, depending on the temperature readings. Another good example for computer scientists familiar with the UNIX operating system are software daemons—background processes which run constantly and react to certain programmed events. A good example of a software daemon is `crond`, a program that reads `crontab` files for each user of the system, and executes commands described in these files periodically.

However, we are naturally more interested in intelligent agents, whose characteristic can be captured by the following three notions [111]:

PROACTIVENESS: intelligent agents take initiative in order to satisfy their goals;

REACTIVITY: intelligent agents adapt to their environment and respond to its changes;

SOCIAL ABILITY: intelligent agents are able to communicate and interact with each other, and possibly form coalitions in order to achieve goals.

It is easy to construct proactive software, we do it all the time when we write functions in Java or C: we define algorithms that produce certain output. This is exactly what proactive behavior is, but it makes an implicit assumption about the environment, namely that it remains unchanged. If it changes, the goal may no longer be achievable, and there is no point in pursuing it any further, and thus we need reactive programs. However, as pointed out by Wooldridge in [109], building reactive software is just as easy: we can devise a reactive algorithm in a straightforward way and make a program that constantly adapts. The difficult part, though, is to create an intelligent agent which knows when to keep pursuing the goal, and when to adapt or stop.

Agent environments

Let us now concentrate on the notion of the environment. Russel & Norvig [88, p. 46] provide the following classification of environment properties:

- *Accessible* vs. *inaccessible:* an agent can either have full and current information about every aspect of the environment or not. Environments which are accessible in this way are usually very simple. The physical world or the Internet are considered inaccessible because of their complexity, despite the fact that they can theoretically be accessible. Generally the more inaccessible the environment, the more difficult it is for the agent to operate in it.

- *Deterministic* vs. *non-deterministic:* in deterministic environments agents can always be certain there is only one possible outcome for each of their actions. While perhaps philosophically controversial, the physical world can be considered non-deterministic. As in the previous point, we will consider any sufficiently complex environment to be non-deterministic as well, even if it actually is deterministic [88].

- *Episodic* vs. *non-episodic:* if an agent has to keep track of the history of its actions throughout different episodes because it may affect its actions, we say it works in an non-episodic environment. In such a situation, agent's behavior may depend not only its current situation, but also has to take into account any previous or future episodes. If there is no link to previous or future episodes, and an agent can only concentrate on a single instance of its task, it works in an episodic environment. A good example of an episodic environment is a mail sorting system [89].

- *Static* vs. *dynamic:* dynamic environments change, static ones can be assumed to remain unchanged.

- *Discrete* vs. *continuous:* environments with a fixed amount of actions can be considered discrete—a games of chess [88] is an example of a discrete environment. A system modeling a taxi ride on the other hand has a continuous environment—its properties (speed, location) change in a constant and continuous way.

The combination of properties that produces the most difficult environment for an agent is an inaccessible, non-deterministic, non-episodic, dynamic and continuous environment [109]. In this thesis we will consider only environments that are accessible, deterministic, episodic, dynamic and discrete, mostly for the purpose of simplicity. The majority of our results can be applied into inaccessible and non-deterministic environments as well, but not to continuous or episodic environments, at least not without significant changes.

Autonomous action

Finally, we discuss the notion of autonomous action. Agents are autonomous in the sense they have no power to control each other's actions, and they are not a subject to any direct, central control. They are free to pursue their goals in whatever way they see fit, as long as the environment allows it. This aspect of intelligent agents is perhaps one that differentiates them most from other software categories, although it introduces more complications to the design of communication protocols [30], but at the same time enables the application of game-theoretic tools, which we will come back to in Section 1.2.2.

Multi-agent systems

So far we have been talking about agents, but not about multi-agent systems. Multi-agent systems, as mentioned in the beginning of this chapter, are systems where there is more than one agent. Taking into account agents' autonomy, environment, their proactive, reactive and social nature,

the study of multi-agent systems gives rise to a vast amount of interesting issues, such as agent communication, coordination, learning, planning, specification and programming. In this thesis we concentrate on coordination problems, and on the foundational aspects of the field: formal logic and game theory.

1.1.1 *Coordination through social laws*

Imagine a situation where a hundred robots are to perform different tasks in a shared environment. Robots are autonomous agents, they can use whatever means necessary to achieve their respective goals. However, their tasks may be conflicting, e. g., one of the robots may take control of a given resource and keep it forever, making it impossible for others to achieve their objectives. Or even a simpler situation may happen, that two robots collide and become damaged. How do we assure that such problems do not occur, i. e., that robots coordinate their actions?

Coordination via limited autonomy

Shoham & Tennenholtz discuss this issue in their seminal paper on social laws [95]. One possible approach is to say that we leave all the coordination in the hands of the robot programmer: his task is to program the robots in such a way that collisions and other problems do not occur. This approach is actually often practiced in robotics [25, 44], but suffers from poor scalability—it is difficult to change the number of robots and given sufficiently many of them it simply becomes a daunting tasks. This approach also sacrifices agent autonomy, as robots are no longer allowed to perform certain actions simply due to their programming.

Coordination via negotiation

The second way of handling the coordination problem is to let agents negotiate with each other. Assuming they are able to communicate, we can implement negotiation and perhaps some special type of agents (schedulers) that would resolve issues that arise when robots cannot come to an agreement. This approach, although studied extensively in AI literature [35, 63, 87], is not flawless either, because if there are no constraints on robots' behavior, negotiations may occur too often, and scheduling agents (if present), may get flooded with the amount of requests.

Coordination via social laws

Thus Shoham & Tennenholtz suggest a solution based on what they call *social laws*.[1] The idea has previously been studied by Moses and Tennenholtz in [70, 71], and it is based on the premise that regulations or laws for artificial agents can be modeled on human societies.

Social laws is an *offline design* paradigm, which means that we assume the presence of a designer that is external to the system. This designer creates the laws: restrictions on agents' behavior that, if abided, guarantee

1 It is worth mentioning the relationship between social laws and *normative systems*. While we sometimes refer to social laws as "norms" or "normative systems", we recognize the latter often refers to a less abstract notion used in deontic logic and communities working on organizations for multi-agent systems. Throughout this thesis we adopt the view that a social law (or a "norm") is nothing more than a behavioral constraint, thus abstracting away from permissions, obligations, and institutions.

that the system exhibits certain desirable properties. For example, given our robot scenario, we could design a law such that robots drive on the right side of the road, thus adhering to right-hand traffic rules. Robots that comply to such law will not run into collisions, yet will not require (as much) negotiation or scheduling and at the same time will remain autonomous.

Social laws framework is not free of problems, though. While the system designer can create optimal laws[2] it is up to the agents to comply to the laws, an important problem known as social law *compliance* (sometimes also referred to as *norm compliance*), which we will revisit in Chapter 2. Another issue is the problem of *feasibility*—is it possible for the agents to achieve their goal(s) if they comply to the laws?—and the related problem of *effectivity*, where by effective social law we understand a law which guarantees the satisfaction of the objective once implemented and obeyed. We will return to these and other social laws-related problems in Chapter 2, but now we discuss the ways we can reason about multi-agent systems.

1.2 REASONING ABOUT MULTI-AGENT SYSTEMS

When we model multi-agent systems for engineering or conceptual purposes, we want to be able to reason about their properties. Can we assure that the system will not have any *deadlocks?* Are we able to determine whether it will exhibit certain desired properties? How about agents, can we say something interesting about their interactions? Do they cooperate? If so, how?

These are the questions we would like answer while designing and analyzing multi-agent systems, and in this thesis we concentrate on using formal logic and game theory to that end.

1.2.1 *Logical approaches*

Using formal logic in computer science can serve three purposes. To quote Van der Hoek and Wooldridge [100], logic can be used for system *specification*, *deduction* and *verification*.

Specification, deduction and verification

By specification we mean a formal description of system's properties. Formal logics can be used in such specification due to their unambiguous, precise mathematical languages, in which many important properties of computer systems can be expressed. Such logical specifications can then be transformed into implementations. But formal logics come with sets of *inference rules* and allow to deduce new facts and properties. These can be used by the system designer if he wants to analyze the system, but may as well be used by the system itself; in multi-agent systems—by agents, who can infer new facts from what they are presented with. Finally, when we

2 What "optimal" exactly means is an open question, see [2] for one possible interpretation.

say that formal logic can be used for verification, we point to *semantics*—sets of rules that specify the "meaning" of logical formulas. Once we have properties of the system specified by such formulas, we can then formally verify whether they are *satisfied* in said system or not.

Model checking

The verification method that became particularly successful in computer science is known as *model checking*. In a model checking problem, we take an abstract representation of the system called (a model) $\mathcal{M}$, a state of this model s and a property expressed as a logical formula φ, and algorithmically check whether φ holds in s of $\mathcal{M}$, formally: $\mathcal{M}, s \models \varphi$. This method, when first applied to temporal logics (which we will talk more about later in this section) by Emerson & Clarke [33], and Quielle & Sifakis [86], became the dominant verification paradigm in computer science and popularized formal logic's usage in the field.

First- and second-order logic

While formal logic usage in computer science is a relatively old idea, initially only propositional calculus, first- and second-order logic were used. These languages served various purposes, from being used in automated theorem proving [46] to specifying logical circuits, SAT-solving [34], and other things. However, it was *modal logic* with its *relational semantics* that proved to be the most flexible and ubiquitous family of logics used in computer science. Modal logic owes its success to a number of factors. Firstly, many modal languages (as we will see in the course of this thesis) achieve good balance between expressive power and complexity of decision problems (most notably model checking), and secondly, modal logics are able to express non truth functional operators which are well suited for capturing notions such as beliefs, desires, knowledge, etc., all of which are desirable in any logical account of agency. The final advantage of modal logic is the conceptual connection between modal languages and computations.

Bisimulation

A key notion in modal model theory is that of *bisimulation*,[3] because modal languages cannot distinguish between models which are bisimilar. The notion of bisimulation is also well known in process algebra, since it is the equivalence relation on process graphs. Thus, bisimilar computations are equivalent, and modal languages are well suited for reasoning about computational processes.

Modal logic

Informally speaking, (propositional) modal logic is a logical language that extends propositional calculus with operators that can express *modal* statements. While there are some controversies as to the exact origin of the first relational semantics, it is often assumed that Saul Kripke was the first to formalize it in a series of papers published in 1960s [64, 65, 66]. Although modal logic had been studied by mathematicians for many years before that,[4] it was Kripke who first gave a formal semantics, which is thus also known as Kripke semantics (or possible worlds semantics).

Relational semantics

In relational semantics formulas of the basic modal language are inter-

3 A bisimulation is a relation between two models where related states have the same atomic properties, and matching transition possibilities.

4 One could, in fact, trace the origins of modal logic back to Aristotle's "De Interpretatione". For more about the complicated history of modal logic, see [19, 67, 62].

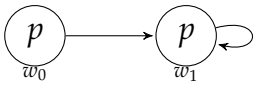

Figure 1.1: A simple Kripke structure with two worlds (or states).

preted over *Kripke structures*, which are tuples of the following shape:

$$K = \langle W, R, V \rangle,$$

where W is a non-empty set of *worlds* (or *states*), $R \subseteq W \times W$ is a binary relation on that set (sometimes called an *accessibility relation*, or a *transition relation* because it can specify how a system can transition from one state to another in transition systems) and $V : W \to 2^{\Phi}$ (where Φ is a non-empty set of propositional symbols)[5] is a *valuation* (or *labeling*) function which to every state $w \in W$ assigns a set of propositions $p \in \Phi$ that is *true* (or *satisfied*) in that state. To give an example of what relational semantics means, we present an inductive definition[6] of the basic modal language with one $\Box$-modality:

Basic modal language

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \Box\varphi,$$

where p is a propositional formula. The intuitive and historical meaning of the $\Box\varphi$ formula is "necessarily p", or in terms of relational semantics: "p is satisfied in all (accessible) possible worlds". We can easily represent Kripke structures as *directed graphs*, with W being the set of *nodes* and R specifying the *edges* of a graph. V then is a function that labels nodes with formulas which are *true* in these nodes. An example of a simple Kripke structure is shown in Figure 1.1. In that structure, the formula $\Box p$ is true in the world w_0, because p is true in every other world that is accessible from w_0 ($\Box p$ is, coincidentally, also true in w_1).

Epistemic logic

In 1962, Jaakko Hintikka published a seminal work on reasoning about knowledge and belief [51] in which he proposed a novel interpretation of the modal connective. His "logic of knowledge" uses $K\varphi$ instead of $\Box\varphi$, which is meant to be read as "φ is known". In this logic, the binary relation on the set of possible worlds is not understood as accessibility, but rather as *indistinguishability*, i.e., if two states are in relation, they are indistinguishable from one another; agents do not *know* which of the states they are in. Hintikka's work, while initially interesting only for a small group of philosophers and philosophically inclined logicians gained a lot

5 The valuation function is often presented in the literature equivalently as $V : W \times \Phi \to \{1, 0\}$.

6 The notation we use throughout the thesis when introducing new formal languages is called a Backus-Naur Form (BNF) notation [15], and is commonly used in formal logic and computer science. A BNF formula for a logical language can be read as follows: φ on the left side of the ::= symbol must be replaced by any of the symbols on the right. Formulas constructed in such way, and only in such way, are called *well formed formulas* of a given logic.

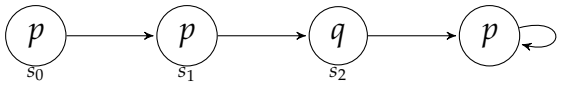

Figure 1.2: A simple LTL model. The formula $\Diamond q$ ("at some point in the future q") is satisfied in the state s_0.

of traction and spawned a whole field known as *epistemic logic*. Currently the field of epistemic logic, formal epistemology and knowledge representation is a huge area of research at the intersection of mathematics, philosophy and computer science. While epistemic logic interpreted the relation between worlds/states as indistinguishability, in *temporal logic* the relation meant the *flow of time*.

Temporal logic

The inspiration came, again, from philosophy, with seminal work of Arthur Prior [83, 84, 85], but it was introduced in the most popular form of Linear Temporal Logic (LTL) by Amir Pnuelli [82]. In LTL, formulas are interpreted over infinite *computations*. These computations are sequences of states in which the progression from one state to another indicates passing of time. A simple example LTL model is presented in Figure 1.2.

LTL

LTL is a *multi-modal* logic, it has more than just one modality: $\bigcirc\varphi$ (or $\mathsf{X}\varphi$), which is read as "in the next state φ is true", $\Box\varphi$ (or $\mathsf{G}\varphi$)—"φ is true always in the future", $\Diamond\varphi$ (or $\mathsf{F}\varphi$)—"φ is true at some point in the future" and $\varphi\mathcal{U}\psi$—"φ is true until ψ becomes true". Using this richer language, LTL is able to express complex properties about computer systems and, thanks to its relational semantics, these properties can be later verified using a *model checking* algorithm. An example of such a property may be

$$\neg\Diamond\Box \mathit{deadlock}, \tag{1.1}$$

which means that it is never the case that eventually the system will end up permanently in a "deadlock" state (this kind of property—"something bad will never happen"—is often called a *safety* property). LTL became so popular that many efficient automatic model checkers have been developed, perhaps the best known of which is SPIN [52], and it has been extensively used in the industry.

LTL gained massive popularity in computer science as it was easy to capture important properties of computer systems, and at the same time complexity of decision problems for LTL is better than for first- or higher-order logics. Model checking the full language of LTL is PSPACE-complete, although for fragments of the language it can be done in time polynomial in the size of the model [97, 17].

CTL

While LTL implements the idea of a linear time line, another popular temporal logic called Computation Tree Logic (CTL) [43] allows to reason about models with *branching time*. CTL has the same temporal modalities as LTL, except each of these modalities is preceded by a connective called a *path quantifier*, which can either be existential: E—"there exists a path",

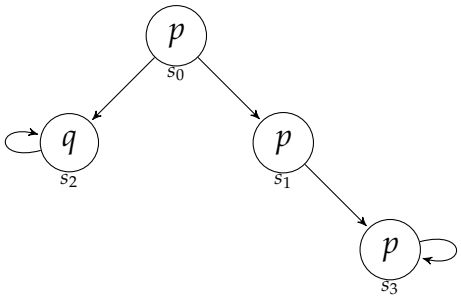

Figure 1.3: A simple CTL model. In state s_0, the formula $\mathsf{E}\bigcirc q$ is true, and the formula $\mathsf{A}\bigcirc p$ is false.

or universal: A—"for all paths". Thus, well-formed formulas of CTL can be captured by the following expression:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{E}\bigcirc \varphi \mid \mathsf{E}\Box\varphi \mid \mathsf{E}\Diamond\varphi \mid \mathsf{E}(\varphi \mathcal{U} \varphi) \mid \mathsf{A}\bigcirc \varphi \mid \mathsf{A}\Box\varphi \mid \mathsf{A}\Diamond\varphi \mid \mathsf{A}(\varphi \mathcal{U} \varphi).$$

An example CTL formula that can be used in software verification is this safety property taken from [54]:

$$\mathsf{A}\Box(\mathit{requested} \rightarrow \mathsf{A}\Diamond \mathit{acknowledged})$$

—for any state, if a *request* occurs it will eventually be acknowledged. Liveness properties can also be easily expressed:

$$\mathsf{A}\Box(\mathsf{E}\Diamond \mathit{restart})$$

—from any state it is possible to get to a *restart* state. An example CTL model is presented in Figure 1.3.

Observe that the grammar defined above does *not* allow for temporal modalities ($\bigcirc$, $\mathcal{U}$, $\Box$ or $\Diamond$) to be used without a corresponding path quantifier. That is why, as pointed out by Ryan & Huth [54], CTL is actually *not* more expressive than LTL, as there are properties expressible in one logic but not in another. A superset of both these languages is known as CTL*, which is a language of CTL without the restriction that a path quantifier must be paired with a temporal modality.

So far we introduced a basic modal language, epistemic logic, and temporal logic. Many other modal logics are used in computer science, deontic logic (logic of obligations and sanctions), intention logics (logics of mental states), but we omit discussing these formalisms, since this thesis is devoted to cooperation, and we are thus more interested in logical approaches to *cooperation*.

There are at least three popular logical frameworks that capture notions of agents' cooperation, strategies and coalitions: Marc Pauly's *Coalition Logic* [76, 77], a family of logics that study the notion of "does-that", commonly known as *seeing-to-it-that* (STIT) logics[7] and *Alternating-time Tempo-*

7 STIT logics have been developed by many authors, but a very good overview can be found in [18, 53].

ral Logic (ATL), developed by Alur, Henzinger and Kupferman [12]. These formalisms are often called *strategic logics*, and it is safe to say that ATL became the most known among the multi-agent systems community.

ATL

ATL takes the idea of a branching-time temporal language introduced in CTL, but adds an additional parameter to its path quantifiers, namely *coalitions* of agents. Here is an example ATL formula:

$$\langle\langle A\rangle\rangle \bigcirc \varphi,$$

and its intended intuitive reading is that a coalition A has a *strategy* to ensure that in the next state φ is the case.

Below we define the complete language of ATL:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\langle A\rangle\rangle \bigcirc \varphi \mid \langle\langle A\rangle\rangle \Box \varphi \mid \langle\langle A\rangle\rangle \Diamond \varphi \mid \langle\langle A\rangle\rangle (\varphi \mathcal{U} \varphi),$$

where p is a propositional symbol and A is a coalition of *agents*. The relationship between ATL's syntax and the language of CTL becomes apparent when we notice that the path quantifier A corresponds to $\langle\langle\emptyset\rangle\rangle$, and the existential quantifier E can be expressed as $\langle\langle\Sigma\rangle\rangle$, where Σ denotes the set of agents.

The truth value of an ATL formula is an outcome of a game played between agents (protagonists) and the system ("environment", antagonist). If the protagonists win, the formula is satisfied, and it means that agents in question have a *winning strategy* for achieving a goal described by the formula. In contrast to the modal logics descibed above (including LTL and CTL), then, ATL formulas are *not* evaluated over standard Kripke structures, but require different models, called *Concurrent Game Structures* (CGSs).

Concurrent Game Structures

Concurrent game structures, as defined in [12], are tuples of the following shape:[8]

$$S = \langle k, Q, \Pi, \pi, d, \delta\rangle,$$

where:

- $k \geq 1$ is number of agents (or *players*),
- Q, Π are finite, non-empty sets of, respectively, states and propositional symbols,
- $\pi : Q \to 2^{\Pi}$ is a labeling function which labels states with (sets of) propositions.
- For every player $i \in \{1, \ldots, k\}$ and each state $q \in Q$, $d_i(q) \geq 1$ is the natural number of available *moves* (or *choices*, or *actions*) for agent i in state q. We identify moves of i at q with the numbers $1, \ldots, d_i(q)$. A *move vector* at q is a tuple $\langle j_1, \ldots, j_k\rangle$, such that $1 \leq j_i \leq d_i(q)$ for every player i. We write $D(q)$ for the set $\{1, \ldots, d_1(q)\} \times \cdots \times \{1, \ldots, d_k(q)\}$ of move vectors, and we call D a *move function*.

8 There are many different equivalent ways to represent concurrent game structures and associated notions such as strategies and outcomes, however throughout this chapter we stick to the notation of [12].

- Finally, for each state $q \in Q$ and each move vector $\langle j_1, \ldots, j_k \rangle \in D(q)$, a state $\delta(q, j_1, \ldots, j_k) \in Q$ is a result of a *transition* from state q if every agent $i \in \{1, \ldots, k\}$ chooses a move j_i. The function δ is called a *transition function*.

ATL computations

We say that q' is a *successor* state of q if there is a move vector $\langle j_1, \ldots, j_k \rangle \in D(q)$ such that $q' = \delta(q, j_1, \ldots, j_k)$, which formally expresses the intuition we conveyed in paragraphs above that the system transitions from one state to another when all agents choose their moves. Additionally, we say that an infinite sequence of states $\lambda = q_0, q_1, q_2, \ldots$ such that for all positions $i \geq 0$ the state q_{i+1} is a successor of q_i is a *computation* of S, and we refer to computations starting at q as q-computations.

ATL formula satisfaction

Let us now formalize the intuitions about the relationship between formula satisfiability and games in ATL. In order to evaluate a formula $\langle\langle A \rangle\rangle \psi$ in a state q of a CGS S, consider the game with an initial position q. To update the position of the game, a protagonist chooses for every agent $i \in A$ a move $j_i \in \{1, \ldots, d_i(q)\}$, and then the antagonist chooses a move $j_b \in \{1, \ldots, d_b(q)\}$ for every agent $b \in \Sigma \setminus A$, where Σ is the set of all agents. The position of the game is then update to $\delta(q, j_1, \ldots, j_k)$. This way a computation is produced, and the protagonist wins if the resulting computation satisfies ψ. The ATL formula $\langle\langle A \rangle\rangle \psi$, then, is satisfied at q if and only if the protagonist has a winning strategy for the game.

ATL strategies

A *strategy* for agent $i \in \{1, \ldots, k\}$ is a function f_i which maps every non-empty finite state sequence $\lambda \in Q^+$ to a natural number such that if the last state of λ is q, then $f_i(q) \leq d_i(q)$. In other words, strategies determine for every finite prefixes of computatations moves for agents. Given a state $q \in Q$, a coalition of agents $A \subseteq \{1, \ldots, k\}$, and a set of strategies $F_A = \{f_a | a \in A\}$, we say an *outcome* of F_A in q is the set $out(q, F_A)$ of q-computations which players in A can enforce if they follow strategies of F_A.

We provided the detailed description of Concurrent Game Structures to emphasize the link between ATL and game theory. One should notice that every time we update the position of the system represented by a CGS, we play normal form games in every state of that structure. We will discuss normal form games in Section 1.2.2 in more detail.

Another important aspect of ATL is the conceptual distinction of agents (players) and the environment. While Kripke structures are well suited for modeling of *closed* systems, where the behavior is fully determined, CGSs are designed to model *open* systems. Alur et al. give a good explanation of what an open system is [12]:

Open systems

> (...) the compositional modeling and design of reactive systems requires each component to be viewed as an open system, where an *open system* is a system that interacts with its environment and whose behavior depends on the state of the system as well as the behavior of the environment. Modeling languages for open systems (...) distinguish between *internal*

> nondeterminism, choices made by the system, and *external* nondeterminism, choices made by the environment. Consequently, besides universal (do all computations satisfy a property?) and existential (does some computation satisfy a property?) questions, a third question arises naturally: can the system resolve its internal choices so that the satisfactions of a property is guaranteed no matter how the environment resolves its external choices?

To give a more intuitive example, ATL is able to express properties of (open) systems that describe what certain agents (or processes) can achieve *regardless of what other agents do*. Recall Formula 1.1. We can reformulate that safety property in the following manner:

$$\langle\langle control \rangle\rangle \Box \neg deadlock,$$

which means that a group of processes called *control* has a strategy that ensures the system will never end up in a *deadlock* state, no matter what other processes do.

What also contributed to ATL's huge popularity within the community was that the computational complexity of the model checking problem is polynomial in the size of the model and length of the formula [12]. Many other variants of ATL were introduced, with epistemic operators [98], with different types of strategies [91, 3, 59], and with different kinds of semantics [80, 78]. Although ATL may seem limited, it proved to be an easily extendable, tractable and flexible logic that is surprisingly expressive.

The above paragraphs describe how modal logic "evolved" into logics of strategic ability, and how different kinds of modalities (belief, knowledge, time, ability) capture many aspects of intelligent agents. But even though this thesis is largely devoted to logical approaches, we also need to mention a different kind of approach to cooperation from which logics for multi-agent systems draw, and that is game theory.

1.2.2 *Game theory*

Game theory is an area of research on the intersection of mathematics, economics, and these days also computer science, that deals with strategies and decision-making. While game theory's initial applications were to the study of economic markets, auctions and in general human interactions, it was quickly discovered that the tools this discipline provides are well suited for multi-agent systems. Here is how Osborne and Rubinstein describe game theory's assumptions in their classic textbook [75]:

> The basic assumptions that underlie the theory are that decision-makers pursue well-defined exogenous objectives (they are *rational*) and take into account their knowledge and expectations of *other* decision-makers' behavior (they *reason strategically*).

If we go back to our definition of intelligent agents from Section 1.1, we will see that a game-theoretic approach to modeling multi-agent systems is a very tempting one.

Games and solutions

Games are abstract models in which *players*[9] are basic entities. A game describes which possible *actions* or *choices* a player can take and his interests, but does not say anything about what players actually *do*. Thus game theory is often concerned with *solutions*, or *solution concepts*, which describe possible *outcomes* of games.

Both games and solution concepts come in many flavors. Probably the most well known games are *strategic games*, also known as *normal form games*, and the most popular solution concept is the *Nash equilibrium*. There are of course other game models and solution concepts, but from a perspective of multi-agent systems research we need to concentrate on the following differences between game models.

Non-cooperative and cooperative games

First of all, we may distinguish *non-cooperative* or *cooperative* games. While this nomenclature is somewhat misleading because it does not do justice to what the true difference between these games is, it does point to the fact that in non-cooperative games individual players are the primitive notion, while cooperative games put groups of players or *coalitions* as the primitive notion. Both these kinds of games have applications to multi-agent systems, but especially recently cooperative games (also known as *characteristic function games*) gained more interest in the community.

Complete vs. incomplete information, perfect vs. imperfect recall

Secondly, games can model players with *complete* or *incomplete* information depending on how much information they are provided about the system, the game or other players' actions, and also with *perfect* or *imperfect recall*, depending on whether we model players that memorize all their previous actions or not (important for *extensive games*). We will return to these distinctions in Chapter 2, and for now we concentrate on what all games have in common.

Rationality

Games in the understanding of game theory adopt a common model of *rational behavior*. In this model, it is assumed that a rational agent has clear preferences, and will choose among available options those that maximize his *utility*, which is understood as a value assigned to each of the possible choices. In formal terms (adopting the notation from [75]), given a set A of available actions, set C of *consequences* of these actions, *a consequence function* $c : A \to C$ and a *preference relation* (a complete transitive reflexive binary relation) $\succsim$ on C, a *rational agent* will choose action b from a set $B \subseteq A$ of *feasible* actions (i.e., those available under certain circumstances), such that $c(b) \succsim c(b')$ for all $b' \in B$. This assumption of what is sometimes called "perfect rationality" is perhaps the most often attacked aspect of game theory, as there is some evidence coming from experimental psychology that contradicts it, but for the purposes of modeling multi-agent systems we find rational agents defined in such way a reasonable model,

9 Since in terms of multi-agent systems agents are players, we will often use these terms interchangeably.

	NOT CONFESS	CONFESS
NOT CONFESS	-1 -1	0 -3
CONFESS	-3 0	-2 -2

Figure 1.4: Prisoner's Dilemma example. Each row of such matrix corresponds to player 1's possible actions, and each column to player 2's. Cells describe outcomes, with player 1's utility on the left.

and throughout this thesis we do not discuss game-theoretic assumptions of rationality.

To make intuitions about game theory clearer, we now briefly discuss two popular models: normal form games (the *de facto* canonical game models) and characteristic function games.

Normal form games

In normal form games all players make one choice and perform actions simultaneously. We say a normal form game is a tuple:

$$G_{norm} = \langle N, (A_i)_{i \in N}, (u_i)_{i \in N} \rangle,$$

where N is a finite set of n players, A_i is a finite set of actions available to player $i \in N$ and $u_i : A \to \mathbb{R}$ is a utility function for player $i \in N$. Additionally, we say that each vector $a = (a_1, \ldots, a_n) \in A$ is an *action profile*.

EXAMPLE 1.2.1: *This is an example coming from [13]. Two prisoners are placed in separate cells (i.e. they cannot communicate with each other) and are offered the following deal: if one of them confesses but the other one does not, they will walk away while the other prisoner gets three years of prison; if they both confess, they both get two years, if neither confesses, they both get one year in prison. An illustration for this example is presented in Figure 1.4.*

Strategies

In game theory we often talk about players' *strategies*. A straightforward type of strategy would be to select one, single action, and play it. This is called a *pure strategy*, and we will refer to such action profiles as *pure strategy profiles*. There are of course other kinds of strategies, like *mixed strategies* (randomizing over choices according to some probability distribution), but we will not discuss other strategies for normal form games in this thesis. We will however discuss solution concepts, particularly the *Nash Equilibrium*.

Nash Equilibrium

Nash Equilibrium [72] (NE) is a solution that captures the notion of *stability*. Informally, it is an action profile where no player can be better off by playing a different action (assuming all other players also play an NE). In other words, no player gains from deviating. Formally, a Nash

Equilibrium for a strategic game is a profile $a^* \in A$ such that for every player $i \in N$,

$$(a^*_{-i}, a^*_i) \succsim_i (a^*_{-i}, a_i)$$

for all $a_i \in A_i$, where $a^*_{-i} = (a^*_j)_{j \in N \setminus \{i\}}$.

Coming back to our Example 1.2.1, we can see that the Nash Equilibrium of this game is playing (Confess, Confess)—a counter-intuitive outcome perhaps.

There are of course other solution concepts and other subclasses of games, which we will go back to in Chapter 2, but for now we want to contrast normal form games with characteristic function games.

Characteristic function games

We mentioned before that one could distinguish non-cooperative and cooperative games, and even though this nomenclature does not reflect the true differences between these classes of games, cooperative games put a coalition of players as a primitive. Thinking along these lines, we say that the normal form games introduced in the previous section are non-cooperative, whereas *characteristic function games* we are about to discuss are cooperative.

A characteristic function game (which we will simply call a cooperative game from now on) is a tuple

$$G_{coop} = \langle N, v \rangle,$$

where N is a finite set of players, and $v : 2^N \to \mathbb{R}$ is a *characteristic function* of the game which assigns a *value* $v(C)$ to each coalition $C \subseteq N$. Note that the characteristic function does *not* say anything about how the value should be distributed amongst the members of a coalition, and it is in fact one of the key research problems in cooperative game theory how this value should be assigned to individual players. There is also an implicit assumption that the value can be distributed among the members of C in any way, i.e., it can be transferred, hence such games are also called *transferable utility games* (TU games, sometimes also called *games with side payments*).

Characteristic functions

EXAMPLE 1.2.2: *We provide a simple example of a cooperative game taken from [28]. Three children, Charlie, Marcy and Pete, have respectively c, m and p amount of dollars. They want to pool their money to buy ice cream, and as is commonly assumed in examples that involve children, they do not assign any utility to money, only to ice cream. There are three sizes of ice cream in the shop: a 500g box for \$7, a 750g box for \$9, and a 1000g box for \$11. This situation can be modeled by a cooperative game with $N = \{C, M, P\}$, and for $c = 3$, $m = 4$, and $p = 5$, the value of a characteristic function v is given by the table below:*

C	$\emptyset$	$\{C\}$	$\{M\}$	$\{P\}$	$\{C, M\}$	$\{C, P\}$	$\{M, P\}$	N
$v(C)$	0	0	0	0	500	500	750	1000

On the other hand, for $c = 8$, $m = 8$ and $p = 1$, the value looks as follows:

C	$\emptyset$	$\{C\}$	$\{M\}$	$\{P\}$	$\{C, M\}$	$\{C, P\}$	$\{M, P\}$	N
$v(C)$	0	500	500	0	1250	750	750	1250

Before we mention solution concepts for cooperative games, we want to recall the Prisoner's Dilemma from Example 1.2.1. As we mentioned while discussing the Nash Equilibrium, the stable strategy for both players is to confess, which is not their best outcome. This is due to the fact that in normal form games *binding agreements are not possible*. If such agreements were possible, the stable outcome would be to play not confess for both players.

There are two kinds of solution concepts for cooperative games:

1. *Fairness*-related solution concepts, which deal with distributing the value of the characteristic function among members of coalitions, and

2. *stability*-related solutions, which, like the Nash Equilibrium in non-cooperative games, deal with finding stable sets of outcomes, i.e., such where no coalition has an incentive to deviate from the coalition structure.

The Core

The most well known stability-related solution concept that is corresponding to Nash Equilibrium in cooperative games is *the core* [47]. Intuitively, the core characterizes a set of outcomes which are stable, such that no coalition has an incentive to deviate. We will return to the discussion about the core and other solution concepts for cooperative game theory in Chapter 2.

1.3 LOGICAL AND GAME-THEORETIC APPROACHES TO SOCIAL LAWS

So far we have discussed the foundational aspects of reasoning about multi-agent systems using formal logic and game theory. In the following section we present the state-of-the-art of research that combines all the elements presented above into formal frameworks for multi-agent systems coordination using the aforementioned social laws paradigm.

Frameworks we will discuss in this section contain three main ingredients:

- a language to express properties (or *goal formulas*) of the system,
- a semantic model in which the properties are evaluated,
- and a restriction on the model, which represents the social law.

Having all these ingredients in place, we are interested in the problems of the following shape: if there is a property of the system expressed in a

given language and interpreted using the given model, are agents able to satisfy these properties if they obey certain restrictions? We would naturally also want to address other questions mentioned in Section 1.1.1.

CTL and agent-labeled Kripke structures

First approaches to the formalization of social laws using modal logic used CTL as a language for objectives and thus, necessarily, Kripke structures to interpret them. However, as we need agents, a simple extension of the Kripke structures is necessary, namely agent labeling. In the same way as we label states with (sets of) propositions true in them, we will now also label edges between these states with agents who "own" them. This sort of a model is usually called an *agent-labeled Kripke structure*. Since the language of goal formulas is CTL, it cannot express agents' strategic abilities, however it is possible to extend CTL's language or reason about social law compliance by other means, i. e. without agents in the object language.

Ågotnes et al. presented this approach in a number of papers. Initially, they developed [4] a logic for reasoning about social laws based on CTL called NTL—*normative temporal logic*—which allowed to express formulas with deontic operators, such as "it is obligatory (resp. permitted) in the context of a social law". Then in [8], the authors study the complexity of a number of decision problems for social laws using CTL and agent-labeled Kripke structures, and in further sections of the paper they extend CTL with *coalition predicates* (similar to those of Quantified Coalition Logic [6]) in order to be able to express social law compliance properties in the object language. Furthermore, in [2], Ågotnes and Wooldridge take the same CTL + agent-labeled Kripke models framework but extend the models with *weights*, in order to model the *costs* of implementing particular law, as well as the fact that the system designer may have differently valued objectives. Thus they study the problem of designing social laws as an *optimization problem*. Again, complexity of designing optimal social law is studied in this paper, and it is also formulated as an integer programming problem.

Optimal social laws

A similar paper from Ågotnes et al. [9], which also studies the problem of social law design, introduces the notion of a *conservative* social law, by which the authors mean the law which makes least changes in the original model, thus being the easiest law to accept and easiest to implement by the designer.

Conservative social laws

Finally, Ågotnes et al. present two papers in which they combine logic and game theory. In both papers agent-labeled Kripke structures are used to model multi-agent systems and CTL formulas represent goals, and social laws are restrictions on the structures, but in [5] authors study normal form games induced by the utility agents assign to different laws (agents also have numerous goals and preferences between them), while in [7] they use cooperative game theory. In the latter paper, authors use the models, social laws and goal formulas to induce simple cooperative games, in which coalitions win when they satisfy a goal while complying to the law and lose otherwise. These games are then used to calculate Banzhaf indices [16] of coalitions—a popular metric to measure players relative *power* in voting systems.

Social laws and games

ATL and Concurrent Game Structures

Since the language of CTL is not able to express agents' strategic abilities, a natural approach would be to use ATL for goal formulas instead. This approach was first presented by Van der Hoek et al. in [102], where authors study all the relevant decision problems mentioned in Section 1.1.1 using ATL and CGSS as models. In this approach, social laws can be designed in a more subtle way, since CGSS allow for specification of actions, and thus a social law can specify which actions are "illegal". A similar approach to social laws using a version of concurrent game structures which satisfy anonymity [80, 78]—*concurrent game structures with roles*—has been presented by Pedersen et al. in [79].

Further reading and references

This chapter is partly based on an excellent textbook edited by Gerhard Weiss [107]. Another great textbook-style position about multi-agent systems in general is a classic position by Michael Wooldridge [108]. A game-theoretic oriented overview to multi-agent systems can be found in a book by Leyton-Brown and Shoham [94], and same authors provide a very short (but excellent) introduction to game theory in [68]. Chalkiadakis, Elkind and Wooldridge on the other hand present a great introduction to computational aspects of cooperative game theory with multi-agent systems in mind in [28]. Finally, a comprehensive, contemporary book about modal logic was published by Blackburn, De Rijke and Venema in 2001 [19] and for a general introduction into logic in computer science look at [54] by Huth and Ryan.

2

PRESENTATION OF MAIN RESULTS

The results of this thesis are presented in the form of four papers that follow this chapter. Here we summarize said results, motivate and discuss their relevance with respect to the state-of-the-art of multi-agent systems research.

2.1 FORMAL REQUIREMENTS FOR MULTI-AGENCY

In Section 1.1 when we characterized agents, we said that they are, amongst other things, autonomous, and then in Section 1.1.1 we mentioned that this property of agents makes coordination of their actions a non-trivial task. Since this thesis' main thread is coordination, before we analyze possible solutions for coordination problems, we want to discuss what are the exact formal requirements for a system to be thought of as multi-agent.

Multi-agent vs. single-agent systems

What exactly distinguishes a multi-agent from a single-agent system? Are systems with multiple agents which execute commands dispatched by a scheduler multi-agent? What is the degree of coordination that is required for agents to be considered autonomous?

Limited coordination

In Paper A, we attempt answering these questions. We claim that true multi-agency requires limited coordination. What we mean by that is that if a coalition of agents is characterized by full coordination on the level of strategies and the information available to its agents, we may as well treat that coalition as a single agent (or a singleton coalition).

Different semantic variants of ATL

As mentioned in Section 1.2.1, one of the most popular formal frameworks for modeling multi-agent systems is ATL, and this is the logic we use for analyzing formal requirements of multi-agency in Paper A. While in the original paper by Alur et al. [12] coordination is given "for free" in the semantics of the logic, many other variants of ATL have been proposed, with different ways of modeling agents' (un)certainty and thus strategies. In an influential paper [91], Schobbens presented four different variants of strategies for ATL based on game-theoretic and epistemic conditions which characterize assumptions behind agents' world view. Can agents see the whole system or just their local state? Do they have any memory of previous states? Recall that in Section 1.2.2 we discussed notions of imperfect- or perfect-information games, as well as imperfect- or perfect-recall—these are exactly the notions incorporated in Schobbens' four variants of strategies for ATL: ATL_{IR}, ATL_{Ir}, ATL_{iR}, and ATL_{ir}, where I (resp. i) stands for perfect (resp. imperfect) information, and R (resp. r) stands for perfect recall (resp. no recall). These distinctions between strategies have ramifications on what level of coordination is present amongst agents, e. g. for imperfect information strategies agents will have the same choices for

the states they cannot distinguish. An element of epistemic logic is introduced here: what does it mean that agents cannot distinguish states in imperfect information strategies? In epistemic logic there a couple of different ways to express collective indistinguishability, and we chose to analyze a variant of strategies in which the members of a coalition can exchange information freely, and thus their collective indistinguishability is an intersection of individual indistinguishability relations—in epistemic logic known as *distributed knowledge*.[1]

In the work presented in Paper A we consider an ATL variant which we call ATL^c_{iy}, where $y \in \{r, R\}$. That is, ATL^c_{iy} is a logic with the language of ordinary ATL, but with strategic modalities interpreted in such a way that their execution is possible under imperfect information (we allow for both no recall and perfect recall strategies). One additional requirement, the "c" in the name of our logic, is that strategies are assumed to be *coalition-uniform*, i. e. for indistinguishable states (or histories in case of perfect recall), all members of a coalition must make the same choice. The main result of Paper A is a theorem that states there exists a truth-preserving translation from the language of ATL^c_{iy} and CGSs into a language we call 1ATL—ATL with only singleton coalitions, *de facto* a single-agent[2] variant of ATL—and its respective models.

Coalition-uniform strategies

This result, although very technical, provides an interesting conceptual insight into the nature of multi-agency in general, and its formal underpinnings in particular. While the pursuit of research conducted in Paper A was philosophical, we believe the results could also be used for computational gains in model checking algorithms for the variant of ATL in question.

2.2 SOCIAL LAWS

When we discussed coordination of multi-agent systems in Section 1.1.1 and 1.3, we mentioned social laws, as a regulatory mechanism somewhat mimicking the laws of human societies. However, we also mentioned the framework comes with a number of challenges, both conceptual and technical.

2.2.1 *The social nature of social laws and compliance*

First there is the conceptual problem of what exactly should a social law forbid. The frameworks mentioned in Section 1.3 based on CTL and Kripke structures, and on ATL and CGSs are fairly simple in this regard: a social

1 As mentioned in the paper, this choice is purely conceptual, i. e. the technical results in the paper hold for other group indistinguishability relations, such as *"everybody knows"* or *common knowledge*.

2 To be exact, the logic of 1ATL is still multi-agent in the sense that there are multiple agents in the system, however the *language* can only express strategic abilities of singleton coalitions, or single agents which represent them.

law is simply a restriction on agents' behavior. In case of agent-labeled Kripke structures, social laws black-list edges, i. e. describe which paths become available or not, depending on the edges' ownership. In CGSS, social laws specify which actions are illegal, i. e. they affect strategic abilities of agents. Neither of these frameworks address what we think is an important *social* aspect of social laws known from human societies, namely that what is forbidden may depend on other agents' actions.

We address this in Paper B, where a novel formalization of a social law (which is referred to as a "norm" in the paper) is presented. We think that simply blacklisting agents' actions is not enough, as compliance to such laws does not require coordination, which seems like an unrealistic scenario. Instead, we think social laws for individual agents which depend on the choices of others model *"normative interaction"*—a concept we find interesting. Compliance to social laws understood in this way might be problematic, as it requires coordination and communication between agents, although if such a situation is impossible, Paper B offers a solution in the form of *collective* social laws, in which we can specify illegal actions not just for single agents but for whole coalitions. "Normative interaction" requires a different notion of compliance. When social laws depend on the action of others, we must demand that a coalition is understood as complying with the law if and only if by doing so it does not prohibit any other agent to comply with that law. Although this notion of compliance is unconventional, we find it interesting and well motivated—again, taking inspiration from laws in human societies.

"Normative interaction"

2.2.2 *Game-theoretic nature of social laws*

Since we mentioned law compliance, we would like to discuss this problem in more detail now.

It is one of the most interesting conceptual problems in the social laws literature: how do we make sure agents comply with the laws? As pointed out in [8], there are basically two ways: we can make laws *incentive compatible* (as in mechanism design [73]), or we can punish agents for non-compliance. These are, however two sides of the same coin, especially when we think of social laws in a game-theoretic way. If we do, we can see that a system consisting of a coalition of agents, a social law and an objective (interpreted over some structure) induces a game in which the coalition is rewarded for achieving the objective when complying to the law and punished for non-compliance. This way the rational thing to do for agents is to comply with the laws. In Paper C we analyze cooperative games similar to those presented by Ågotnes et al. in [7], but while these authors concentrated on using induced games to calculate Banzhaf indices of players, we concentrate on other game-theoretic properties and solution concepts.

Dealing with non-compliance

In essence, our games reward coalitions for the amount of goals they can achieve while complying to the laws *minus* the amount of goals they

Compliance games

could achieve without compliance. The unusual design of our games is intended as a guide to the system designer: if agents can achieve more goals while ignoring the laws than while complying then either the system, the objectives or the laws are not optimal. Apart from that we are able to reason about social laws in novel ways, e. g. we can ask about non-emptiness of the core. The core, as mentioned in Section 1.2.2, is the solution concept that captures the notion of stability in cooperative games, and characterizes a set of outcomes in which no coalition has an incentive to abandon current coalition structure. This particular solution concept has an intriguing and non-standard interpretation in case of our games. Due to the semantics of the (partial) compliance operator of the underlaying formal framework, if a coalition stays in a given coalition structure and gets some value that makes them want to remain in this structure (i. e. the players will not be better off teaming up with others), it means at the same time that we are rewarding this coalition for the *hypothetical* situation in which it is complying to the law and *no other coalition is*. This is because we are interested in scenarios where we do not know anything about players other than the complying coalition, and thus we assume the worst case, that is, that other players discard the law. The core, then, is a set of such hypothetical scenarios in which we identify coalitions which are better off by complying while no other players comply. An empty core would indicate a design flaw in the system, the laws or the objectives.

Core of compliance games

However, game-theoretic nature of social laws can not only be used for conceptual purposes, but also for computational gain. In Paper B, we observe that there is a connection between Nash Equilibria of *anonymous games* induced by social laws and goal formulas and the set of legal actions for players. We then use this observation to prove that all such induced games belong to a class of normal form games which admits a compact representation, and can thus have their Nash Equilibria computed in polynomial time. In effect, we are able to reduce the complexity of some decision problems for the logical language we use in Paper B.

Anonymous games

2.2.3 *Computational problems*

Since we mentioned how anonymous games aid in reducing complexity of finding Nash Equilibria, we want to discuss computational complexity of decision problems for formal frameworks that implement social laws.

Most results for computational problems in the social laws literature are negative. Finding a coalition C whose compliance to a law is *necessary* (resp. *sufficient*)—C-necessity (resp. C-sufficiency)—for the objective to hold is co-NP-complete for CTL and Kripke models [8]. Problems of law feasibility or effectivity are also generally hard in all the frameworks, and one would expect our solutions from Paper B to be even harder to compute for due to the expressive power of our "interactive" norms, however that is not the case.

The way we avoid paying extra computational costs is by using *homogeneous* concurrent game structures. This class of semantic structures for ATL was introduced by Pedersen et al. in [80] and later axiomatized in [78]. For this type of structures we impose an additional axiom for ATL which says that strategic abilities of coalitions of equal size are the same:

Homogeneous Concurrent Game Structures

$$\langle\langle C\rangle\rangle \bigcirc \varphi \text{ iff } \langle\langle D\rangle\rangle \bigcirc \varphi \text{ if } |C| = |D|.$$

The underlying structure does not distinguish the identity of agents, only their number (that is the reason why we employed anonymous games as an incentive mechanism), however the social laws that we introduce in Paper B, both individual and collective, do *not* carry this homogeneity restriction, and are thus used to regain the expressive power we lose by using homogeneous structures. At the same time we extend the language of ATL by an extra element, $\langle C\rangle\varphi$, which is a law *compliance* operator, and its intuitive reading is that the formula φ holds if coalition C complies to some social law χ given as a parameter. This extended language, as we prove in Paper B, admits a model checking algorithm which is polynomial in the number of agents (however, other elements of the system—states and actions—must remain constant), and this results hinges on the correspondence between anonymous games mentioned in the previous section and sets of legal actions available to complying agents in our framework. We thus find the resulting logic suitable for reasoning about social law compliance in systems with a large number of agents.

There are unfortunately also some negative computational results we present in Paper C. While analyzing properties of the characteristic function of compliance games introduced in that paper we observe that an arbitrary characteristic function can induce a compliance game and there are no interesting properties we can take computational advantage of. As a corollary, we show that this result holds also for cooperative games introduced by Ågotnes et al. in [7]. In effect, all the interesting computational problems for compliance games (core non-emptiness, finding dummy players, calculating Shapley values) remain intractable in the general case.

Compliance games intractability

2.3 IMPLEMENTING A NORM COMPLIANCE MODEL CHECKER

So far the results we presented are of a conceptual and technical nature, and in general very abstract and theoretical. However, as we advertise multi-agent systems' growing influence in systems design and engineering in Section 1.1, it is only fair that we show how the social laws paradigm can be applied, and to that end we present a *norm compliance temporal logic model checker* in Paper D.

The model checker presented there, NorMC, is written in the Haskell programming language [60], and is designed for the *Norm Compliance* CTL (NCCTL) introduced in [8]. The main motivation behind writing such a tool was to explore how well is Haskell suited for designing logic-related software like model checkers or theorem provers, and also to create a model

NorMC model checker

checker capable of handling *model update semantics* featured in NCCTL. At the time of writing the paper, as far as we are aware, there was only one model checker that supported this kind of semantics, DEMO [103], but it was designed for epistemic and not temporal logics. Other model checkers for branching-time temporal logics did not support model updates and it was difficult (or impossible) to extend them.

Our research provided some interesting insight into Haskell's suitability for logic-related software. The language proved to be very succinct, easily readable and relatively close to discrete mathematics (as also argued by authors of [40] and [74]). Indeed, as a purely functional language, Haskell is perfectly suited for expressing logical formulas, recursive algorithms, etc. In fact we present formal definitions of structures in the paper and in the code, and very often there are only slight differences. We believe this contributes to the code being easy to understand for not only computer scientists unfamiliar with model checking or Haskell itself, but also for analytically inclined philosophers interested in the social laws paradigm.

Despite the relatively small code-base which allowed for only several optimizations, the code proved to be surprisingly efficient, due the excellent Glasgow Haskell Compiler. Additionally, NORMC proved to be flexible and easily extendable, because the language for describing models, objectives and social law is pure Haskell, thus, in contrast to other model checking tools which use specific modeling languages, we can employ the power of a full programming language to easily implement complex scenarios and, if necessary, extend the model checker itself.

Finally, we chose Haskell because in contrast to niche academic languages it is a general-purpose programming language with a lively community, excellent documentation and growing industrial usage.

Paper D also points to an online repository which contains an up-to-date version of our tool published under the terms of BSD license.

3

CONCLUSIONS

We would now like to discuss the main contributions of this thesis, relate it to other areas of research in multi-agent systems and outline plans for possible future work. As mentioned before, the main thread of the thesis is coordination of multi-agent systems. We have analyzed coordination on three different levels here: conceptual, technical and practical.

Conceptual analysis

The conceptual analysis we presented sheds new light on the formal requirements for multi-agency from a logical perspective, and provides insight into the technical aspects of how coordination is implemented within the most popular strategic logic in the community, ATL. We have shown that with full communication on the level of strategies, multi-agency boils down to single-agency understood as "single-agent *vs.* the environment".

This thesis advocates the social laws paradigm as a formal framework for coordination, and we presented a number of conceptual results regarding that particular framework as well. Our general conceptual contribution here is emphasizing the game-theoretic and social nature of social laws. We presented a novel approach which formalizes social laws which are dependent on actions of other agents, and we have presented new kind of cooperative games induced by social laws.

Technical contributions

Each conceptual analysis conducted in this thesis was backed up by formal, technical results. We have presented a translation from a variant of ATL with imperfect information, coalition-uniform strategies into a language with only singleton coalitions and proved that it is indeed a truth-preserving translation. We have also shown how anonymous games and homogeneous concurrent game structures can be used to reduce the complexity of model checking for a logic that employs our new notion of social laws which depend on actions of other agents, and proved that it can indeed be done in time polynomial in the number of agents, for the full language of ATL extended with law compliance operators. Finally, we have provided a detailed technical analysis of cooperative games induced by social laws with a representation theorem for characteristic functions of said games and complexity results for computing some solution concepts.

Practical contribution

All the work mentioned above was purely theoretical, however since the motivation for multi-agent systems research comes mainly from computer engineering, we also presented a practical contribution in the form of a prototype model checker, NorMC, which, to the best of our knowledge, is the only model checking tool available for verification of social law compliance properties.

We conclude that the interactive nature of social laws deserves further investigation and seems like an intriguing concept. It remains abstract as a formal framework, at the same time adding more subtlety to the notion

of a social law yet not causing computational problems. We also conclude that the game theoretic approaches to social laws are a promising area of research, as social laws seem to naturally induce cooperative games and we believe we have shown that cooperative game theory tools which we can borrow are quite powerful. Finally we found that the Haskell programming language is indeed perfectly well suited for model checking social laws and law compliance; we have shown it is an easy to use, robust, flexible and efficient language.

Results presented in this thesis come from four papers, three of which have been published or accepted for publishing, and one of which (Paper C) is still under review at the time of writing. The three published papers have gone through peer-review process of the international community of multi-agent systems researchers and have been presented during international conferences.

3.1 FUTURE AND RELATED WORK

In all the sections above we referenced background and state-of-the-art material which was related to the work we presented, however we would now like to also relate our results to research outside the field of multi-agent systems. We hope that one of the contributions of this thesis is to open up research in several directions.

Social science

The most natural link we can draw between multi-agent systems coordination is social science, since as we mentioned before, our own research is in many places inspired by the study of human societies. Game-theoretic notions that we often use, such as strategies, equilibria, stability etc., are very commonly used in economics literature, to study both micro- and macro-economic phenomena. In addition, the incentive-based social laws design that we studied here is closely related to *mechanism design* [73], a subfield of game theory which is intensively studied in economics. As a matter of fact the 2007 Nobel Memorial Prize in Economic Sciences was awarded to mechanism design inventors, Leonid Hurwicz, Eric Maskin and Roger Myerson, and the 2014 prize recipient, Jean Tirole, also used mechanism design for the analysis of market power and regulation.

Another and perhaps closer field related to our research is that of *computational social choice* [23], a field which uses algorithmic methods to study coalition formation, resource division, voting theory and preference aggregation. Our study of social laws is particularly closely related to coalition formation problems, as we are interested in the ways agents join coalitions of one shape and not the other, but we are also interested in voting theory since we employ cooperative games.

Software verification

On the technical front our work is related to software verification in computer science, since we study model checking problems and algorithms, and we are interested in optimizing the complexity of said problems. While we study coordination, we are particularly interested in properties of multi-agent systems that we can express in the language of some

logic, in our case CTL or ATL, and then verify whether the system satisfies these properties—that is precisely the topic of interest of the software verification community. As there is a strong link between multi-agent systems and software verification research areas, we believe that some of our technical contributions in model checking may be found useful in software verification.

Algorithmic graph theory

At the same time we note there is a close relation between our research and the field of algorithmic graph theory, since all the semantic models we consider for our logics are in fact (directed) graphs with different kinds of labels, and our model checking problems are closely related to graph's analysis. Indeed even the game theoretic approaches to social laws that we presented here give rise to algorithmic problems on graphs, and there is a strong relationship between efficient representation of cooperative games and graph theory.

Future work

When it comes to future work, we think there is a lot of interesting topics to pursue in every area of our contributions, with a particular emphasis on the link between game theory and social laws. There are more classes of games that can be induced by social laws that should be studied, especially non-transferable utility games which we have not analyzed in this thesis. Furthermore, the link between mechanism design and social laws should be investigated more, e. g. we would like to know if social laws design could be represented as a social choice process. We would also like to further investigate the notion of "normative interaction" and analyze other variants of it, e. g. systems that take into account whether complying to a certain law makes it easier/harder for other agents to comply, and if so, how much easier/harder. Finally, in any work that employs formal logic and algorithmic game theory there is always room for improving the complexity of decision problems, and if theoretically impossible then for finding tractable instances, which is especially interesting in case of compliance games that we introduced here.

Part II

PAPERS

4

PAPER A: MULTI-AGENCY IS COORDINATION AND (LIMITED) COMMUNICATION

This paper was accepted for a presentation during the 17th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA 2014), taking place on December 1–5, 2014 in Gold Coast, Queensland, Australia. The paper was accepted for publication in the conference proceedings by Springer in the Lectures Notes in Artificial Intelligence (LNAI) series.

MULTI-AGENCY IS COORDINATION AND (LIMITED) COMMUNICATION

Piotr Kaźmierczak
Dept. of Computing, Mathematics and Physics
Bergen University College, Norway

Thomas Ågotnes
Dept. of Information Science and Media Studies
University of Bergen, Norway

Wojciech Jamroga
Computer Science and Communication, and Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg
Institute of Computer Science, Polish Academy of Science, Poland

Abstract

Systems within the agent-oriented paradigm range from ones where a single agent is coupled with an environment to ones inhabited by a large number of autonomous entities. In this paper, we look at what distinguishes single-agent systems from multi-agent systems. We claim that *multi*-agency implies limited coordination, in terms of action and/or information. If a team is characterized by full coordination both on the level of action choice and the available information, then we may as well see the team as a single agent in disguise. To back the claim formally, we consider a variant of Alternating-time Temporal Logic ATL where each coalition operates with a single indistinguishability relation. For this variant, we propose a truth-preserving translation of formulas and models in the syntactic fragment of ATL where only singleton coalitions are allowed. In consequence, we show that assuming unified view of the world on part of each coalition reduces the full language of ATL to its single-agent fragment when a model is given.

1 INTRODUCTION

Agent-based models become a suitable foundation for IT environments nowadays. More and more systems involve social as much as technological aspects, and even those that focus on technology are often based on distributed components exhibiting self-interested, goal-directed behavior. Moreover, the components act in environments characterized by incom-

plete information and uncertainty. The field of *multi-agent systems* [108] studies the whole spectrum of phenomena ranging from agent architectures to communication and coordination in agent groups to agent-oriented software engineering. The theoretical foundations are mainly based on game theory and formal logic.

Systems within the agent-oriented paradigm display various degrees of multiplicity, from systems where a single agent is coupled with an environment (often used e.g. in agent-oriented programming), to massively populous ones used e.g. for agent-based simulation. What distinguishes a single-agent system from a *multi*-agent system is an interesting question in itself. In particular, is it enough that a system consists of multiple *modules* to call it multi-agent? What about semi-autonomous entities that act according to "orders" dispatched from a central unit? Or entities that act autonomously but they pursue a common goal, and act according to a joint plan? All these cases clearly display different degrees of autonomy and agency.

In this paper, we claim that multi-agency implies limited coordination, in terms of action and/or information. That is, different agents may collaborate, but they are inherently separated: each agent is individually responsible for executing his/her actions, and does that based on his/her individual view of the situation. Putting it in another way, if a team is characterized by full coordination both on the level of action/strategy choice and the available information, then we may as well see the team as a single (though composite) agent.

To back the claim formally, we use Alternating-time Temporal Logic ATL [11, 12] that combines elements of game theory, temporal logic, and epistemic logic in a neat formal framework. Coordination of coalitional strategies is implicitly given "for free" in the semantics of ATL, but the logic has many semantic variants for reasoning about coalitional play under different patterns of uncertainty. We consider a variant of ATL where each coalition operates by definition with a single indistinguishability relation (e.g., the distributed knowledge relation). For this variant, we propose a truth-preserving translation of formulas and models in the syntactic fragment of ATL where only singleton coalitions are allowed. In consequence, we show that assuming unified view of the world on part of each coalition reduces the full language of ATL to its single-agent fragment, at least when a model is given.

The main purpose of this study is philosophical. We want to understand the different degrees of autonomy and agency that arise in complex systems. Still, the reduction that we propose can be potentially used to implement model checking for some interesting semantic variants of ATL.

We begin by introducing the relevant syntactic and semantic variants of ATL in Sections 2 and 3. Then, we present our main result in Section 4. We conclude with some final remarks in Section 5.

RELATED WORK. ATL has been studied extensively in the last 15 years. The research can be roughly divided into the computational and conceptual strands. The conceptual strand focuses on looking for the "right" semantics of ability, especially in the presence of imperfect or incomplete information. ATL has been combined with epistemic logic [99, 104, 1, 56], and several semantic variants were defined that match various possible interpretations of strategic ability [91, 59, 56]. Multiple extensions have been considered, e.g., with explicit reasoning about strategies, rationality assumptions and solution concepts [105, 101, 106, 29], agents with bounded resources [10, 27], coalition formation and negotiation [26], opponent modeling and action in stochastic environments [55, 90] and reasoning about irrevocable plans and interplay between strategies of different agents [3, 24]. Besides providing a palette of different formal interpretations for the concept of strategic ability, the research brought benefits in analysis of related verification problems, such as module checking [58].

In this paper, we are especially interested in works that redefine the "uniformity" conditions for coalitional play, based on a *single* epistemic relation for the whole coalition [48, 39, 49, 37]. Philosophically, this amounts to assuming members of the coalition to share their knowledge (or, conversely, propagate their uncertainty) at each step while executing a joint strategy. We show that – at least in the context of model checking – such a coalition can be seen as a single agent executing compound actions *in unison*.

2 REASONING ABOUT ABILITIES OF AGENTS AND COALITIONS

Alternating-time Temporal Logic ATL [11, 12] is a non-normal modal logic that allows for expressing properties of multi-agent systems. Specification in ATL is usually based on formulae of type $\langle\langle A\rangle\rangle\varphi$, expressing that the group of agents A has a strategy to enforce the temporal property φ no matter what the other agents do. Formulae of ATL are interpreted in *concurrent game structures* that assume synchronous execution of actions from all the agents in the system. We begin by defining the models formally. Then, we present the syntax and the semantic clauses for relevant variants of the logic.

2.1 *Models: Concurrent Game Structures*

In the most general case, formulas of ATL are interpreted over *imperfect information concurrent game structures* (ICGS), defined as follows:

DEFINITION 2.1 (IMPERFECT INFORMATION CONCURRENT GAME STRUCTURES [91]): *An* ICGS *is an 8-tuple* $M = \langle \Sigma, \Pi, Q, C, d, \delta, \sim, \pi\rangle$, *where:*

- Σ *is a finite set of agents;*
- Π *is a finite set of propositional letters;*

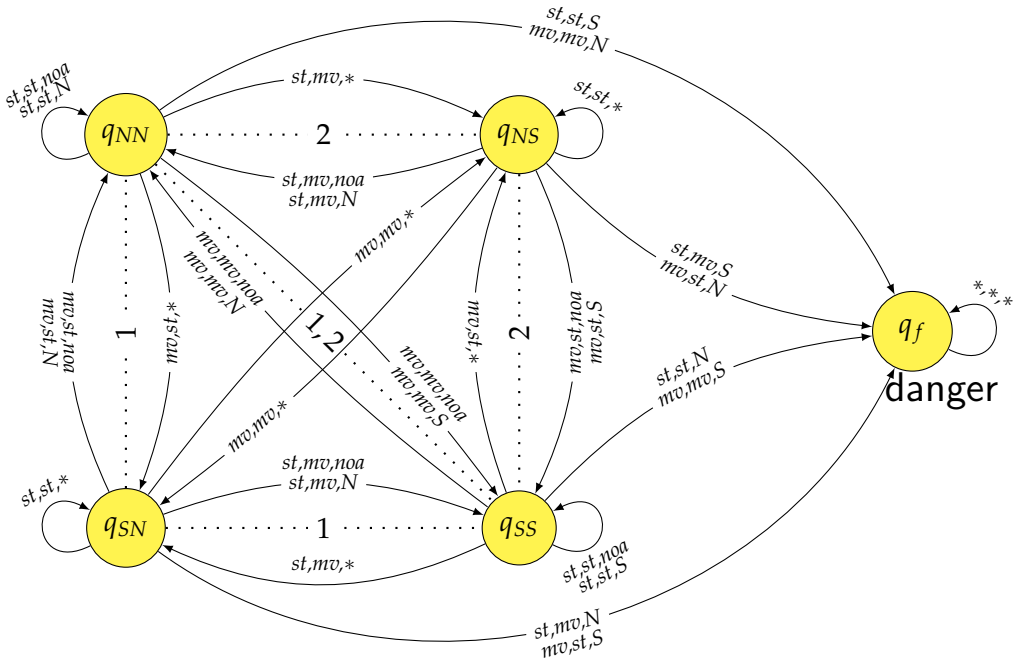

Figure 4.1: A simple model of coordinated defense (M_1). The wildcard ($*$) matches any action of the appropriate player

- Q *is a finite set of states;*
- C *is a finite set of choices/actions;*
- $d : Q \times \Sigma \to 2^C$ *is a* guard function *that specifies which actions are enabled for whom and where. It is assumed that in any state at least one choice must be enabled for each agent. We will usually write* $d_a(q)$ *instead of* $d(q, a)$*;*
- $\delta : Q \times C^\Sigma \to Q$ *is a deterministic transition function;*
- $\sim : \Sigma \to 2^{Q \times Q}$ *is a family of equivalence relations (one per agent) indicating states that are indistinguishable from agents' perspectives, and*
- $\pi : Q \to 2^\Pi$ *is a valuation of atomic propositions.*

Additionally, it is usually assumed that the ICGS *assigns the same sets of choices to indistinguishable states; formally: if* $q \sim_a q'$ *then* $d_a(q) = d_a(q')$.

We illustrate how ICGS are used to model multi-agent systems on the following example.

EXAMPLE 2.1 (COORDINATED DEFENSE): *Two guards (agents* 1 *and* 2*) are supposed to protect a sensitive area from attack. They conduct surveillance of the area in parallel, from two separate locations. These can be different floors in a building, or different hills giving view to a military zone, etc. At any moment, each guard is in a position that allows him to protect either the North or the*

South entry to the area, but not both at the same time. Moreover, a guard can stay in the same place (action st) or move to the other side of the surveillance area (action mv). However, the landscape is confusing and the guards are no experts in reading landscape signs; in consequence, both entries and surveillance points look the same to each guard. On the other hand, guard 1 can recognize when he is in the North position and guard 2 is in the South position, because only then he can see the light from the other guard's torch. Likewise, guard 2 can only distinguish the situation when he is in the North and the other guard is in the South.

The attack – executed by the third agent a, the "attacker" – can be conducted either from the North (action N) or from the South (action S). The attacker can also refrain from attacking (action noa). The attack is only successful if it targets a position which, in the very next moment, will not be protected by any of the guards. In such case, the system proceeds to the "failure" state q_f, labeled by the atomic proposition danger.

A simple model of the scenario is depicted in Figure 4.1. The set of players is $\Sigma = \{1, 2, a\}$. States, transitions (represented by solid arrows), indistinguishability relations (represented by dotted lines), and valuation of atomic propositions can be easily read off the picture.

2.2 *Syntax: ATL and Single-Agent ATL*

The language of alternating-time temporal logic, formally referred to as $\mathcal{L}_{\text{ATL}}$, is defined by the following grammar:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A\rangle\rangle \bigcirc \varphi \mid \langle\langle A\rangle\rangle \Box \varphi \mid \langle\langle A\rangle\rangle \varphi \,\mathcal{U}\, \varphi$$

where p is a propositional symbol and A is a subset of agents (called sometimes a *coalition*). We will write $\langle\langle a_1, a_2, \dots\rangle\rangle$ instead of $\langle\langle\{a_1, a_2, \dots\}\rangle\rangle$. The temporal operator $\bigcirc$ stands for "in the next moment", $\Box$ for "always from now on", and $\mathcal{U}$ for "strong until". We use the usual abbreviations of Boolean operators, plus the standard abbreviation for "sometime in the future": $\Diamond\varphi \equiv \top\,\mathcal{U}\,\varphi$.

We also define a syntactic fragment of ATL called the *single-agent* ATL, that allows only singleton coalitions in the formulae. The fragment, to which we will refer as $\mathcal{L}_{\text{1ATL}}$, is formally defined as follows:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle a\rangle\rangle \bigcirc \varphi \mid \langle\langle a\rangle\rangle \Box \varphi \mid \langle\langle a\rangle\rangle \varphi \,\mathcal{U}\, \varphi$$

where p is a propositional symbol and a is an agent.

EXAMPLE 2.2: *The following formulae specify possible requirements on the Coordinated Defense scenario:* $\langle\langle 1, 2\rangle\rangle \Box \neg$danger *(the guards can effectively protect the system from attacks forever),* $\neg\langle\langle 1, 2\rangle\rangle \Diamond \langle\langle 1, a\rangle\rangle \bigcirc$ danger *(the guards cannot compromise the system in such a way that, at some future moment, guard 1 can collude with the attacker for a successful attack).*

2.3 *Semantic Variants of ATL*

Semantic variants of ATL are usually derived from different assumptions about agents' capabilities. Can the agents "see" the current state of the system, or only a part of it? Can they memorize the whole history of observations in the game? Different answers to these questions induce different semantics of strategic ability. In this section, we recall the "canonical" variants as proposed in [91]. There, a taxonomy of four strategy types was introduced and labeled as follows: I (resp. i) stands for *perfect* (resp. *imperfect*) *information*, and R (resp. r) refers to *perfect recall* (resp. *no recall*). In essence, the semantics of ATL in [91] is parameterized with the strategy type – yielding four different semantic variants of the logic, labeled accordingly (ATL_{IR}, ATL_{Ir}, ATL_{iR}, and ATL_{ir}).

The following types of strategies are used in the respective semantic variants:

- Ir: $f_a : St \to Act$ such that $f_a(q) \in d(a, q)$ for all q;
- IR: $f_a : St^+ \to Act$ such that $f_a(h) \in d(a, q_n)$ for all $h = q_0, \ldots, q_n$;
- ir: like Ir, with the additional constraint that $q \sim_a q'$ implies $f_a(q) = f_a(q')$;
- iR: like IR, with the additional constraint that $h \approx_a h'$ implies $f_a(h) = f_a(h')$, where $h \approx_a h'$ iff $h[i] \sim_a h'[i]$ for all i.

That is, strategy f_a is a conditional plan that specifies agent a's actions in each state of the system (for memoryless agents) or for every possible history of the system evolution (for agents with perfect recall). Moreover, imperfect information strategies specify the same choices for indistinguishable states (resp. histories). Finally, a collective xy-strategy F_A for a group of agents $A \subseteq \Sigma$ is a tuple of xy-strategies $(f_a)_{a \in A}$, one for each agent.

A *computation* is an infinite sequence of states $\lambda = q_0, q_1, \ldots$, and we say it is an *outcome* of strategy F_A from state q if $q_0 = q$ and for each $i > 0$, there are $c_a \in C$ choices for $a \in \Sigma \setminus A$, such that $q_{i+1} = \delta(q_i, \mathbf{c})$ where $\mathbf{c}_a = f_a([q_i]_a)$ if $a \in A$, and $\mathbf{c}_a = c_a$ for the opponents. Outcomes are therefore computations that start from the given state and follow the strategy. The set of all the outcome paths of strategy F_A from state q on is denoted by $out(q, F_A)$.

Given an ICGS M, state q and formula φ, we interpret the formula as follows:

- $M, q \models_{\text{xy}} \langle\langle A \rangle\rangle \bigcirc \varphi$ iff there exists an xy-strategy F_A for A such that, for every q' with $q \sim_A q'$, and every $\lambda \in out(q', F_A)$, we have $\lambda[1] \models_{\text{xy}} \varphi$.
- $M, q \models_{\text{xy}} \langle\langle A \rangle\rangle \Box \varphi$ iff there exists an xy-strategy F_A for A such that, for every q' with $q \sim_A q'$, and every $\lambda \in out(q', F_A)$, we have $\lambda[i] \models_{\text{xy}} \varphi$ for every position $i \geq 0$.

- $M, q \models_{xy} \langle\langle A \rangle\rangle \varphi_1 \mathcal{U} \varphi_2$ iff there exists an xy-strategy F_A such that, for every q' with $q \sim_A q'$, and every $\lambda \in out(q', F_A)$, there is a position $i \geq 0$ such that $\lambda[i] \models_{xy} \varphi_2$, and for all positions $0 \leq j < i$ we have $\lambda[j] \models_{xy} \varphi_1$.

In the above clauses, one element is not properly defined yet – namely, the coalitional indistinguishability relation $\sim_A$. Epistemic logic suggests several "canonical" ways in which collective indistinguishability can be constructed. The epistemic relation for *"everybody knows"* is defined as the union of individual relations: $\sim_A^E = \bigcup_{a \in A} \sim_a$. The relation for *common knowledge* ($\sim_A^C$) is the transitive closure of $\sim_A^E$. Furthermore, the epistemic relation for *distributed knowledge* is defined as the intersection of individual relations: $\sim_A^D = \bigcap_{a \in A} \sim_a$. Since we focus on coalitions that can freely communicate and exchange information, we assume that $\sim_A = \sim_A^D$. Notice that the distinction is not relevant for the main result in this paper, which proceeds by embedding coalition-uniform abilities in 1ATL. This is because, for individual agents, $\sim_{\{a\}}^D = \sim_{\{a\}}^E = \sim_{\{a\}}^C = \sim_a$.

We illustrate how the semantics works on the Coordinated Defense example.

EXAMPLE 2.3: *For model M_1 from Example 2.1 and formulae from Example 2.2 we have the following. $M_1, q_{SN} \models_{ir} \langle\langle 1,2 \rangle\rangle \Box \neg$danger because (i) the coalition $\{1,2\}$ has distributed knowledge that the initial state is precisely q_{SN} and (ii) from q_{SN}, the memoryless strategy where each guard does st in every state avoids q_f no matter what the attacker does. On the other hand, $M_1, q_{NN} \not\models_{ir} \langle\langle 1,2 \rangle\rangle \Box \neg$danger because the only way to avoid a successful attack right after the game begins is that one guard stays, and one moves to the other position. Since they use memoryless strategies, the staying guard must execute st forever. Moreover, the moving guard will not see that he has changed his position (as $q_{NN} \sim_1 q_{SN}$ and $q_{NN} \sim_2 q_{NS}$). Since they can only use uniform strategies, the moving guard must execute mv forever – but that means that the area can be successfully attacked in two steps from the start. Finally, memory matters: $M_1, q_{NN} \models_{iR} \langle\langle 1,2 \rangle\rangle \Box \neg$danger. To see this, consider any strategy where one guard always stays, and the other one moves in the first moment, and stays from then on. We leave it up to the reader to check that the strategy succeeds from $\{q_{NN}, q_{SS}\}$, i.e., both states that the guards jointly consider possible at the beginning.*

As a *logic*, we will understand the language together with the chosen semantic interpretation. For the logics used in this paper, we will use the notation $L_{xy} = (\mathcal{L}_L, \models_{xy})$. For example, 1ATL_{ir} denotes the logic with the syntax defined by $\mathcal{L}_{1\text{ATL}}$ and the semantics by $\models_{\text{ir}}$.

3 A DIFFERENT CONCEPT OF COALITIONAL UNIFORMITY

The uniformity conditions presented in Section 2.3 are based on the assumption that each member of the coalition executes its part of the joint

plan on its own. Thus, the execution of every next step is based on the agent's individual view of the situation. A number of papers redefine uniformity of coalitional strategies, using instead a *single* epistemic relation for the whole coalition [48, 39, 49, 37]. This amounts to assuming members of the coalition to establish their *joint view of the situation* at each step while executing the joint strategy. Thus, at each step they either fully share their individual knowledge, or aggregate their uncertainty. [48, 39, 49] take the first approach by defining coalitional uniformity on top of the distributed knowledge relation. In [37], the opposite stance is adopted, by assuming that members of a coalition must choose same actions in states that are connected by the common knowledge relation. The main motivation for the semantic variations was quest for a variant of ATL with imperfect information, perfect recall, and decidable model checking.[1] However, the research was conceptually interesting in its own right.

We formalize the intuitions from [48, 39, 49, 37] by changing the set of available strategies as follows.

DEFINITION 3.1 (COALITION-UNIFORM STRATEGIES): *A collective memoryless strategy f_A is* coalition-uniform *iff $q \sim_A q'$ implies $f_a(q) = f_a(q')$ for every $q, q' \in St$ and $a \in A$. Likewise, a collective perfect recall strategy f_A is* coalition-uniform *iff $h \approx_A h'$ implies $f_a(h) = f_a(h')$ for every $h, h' \in St^+$ and $a \in A$.*

Note that uniformity constraints are relevant only for the imperfect information case. Depending on the type of recall, we will denote the new variant of ATL by ATL^c_{ir} or ATL^c_{iR}, with "c" standing for *imperfect information with coalition-uniform strategies*. Let $y \in \{r, R\}$. The interpretation of strategic modalities in ATL^c_{iy} is defined below:

- $M, q \models^c_{\text{iy}} \langle\langle A\rangle\rangle \bigcirc \varphi$ iff there exists a set of coalition-uniform y-strategies F_A, such that for all q' with $q \sim_A q'$ and all computations $\lambda \in out(q', F_A)$ we have $\lambda[1] \models^c_{\text{iy}} \varphi$.
- $M, q \models^c_{\text{iy}} \langle\langle A\rangle\rangle \Box \varphi$ iff there exists a set of coalition-uniform y-strategies F_A, such that for all q' with $q \sim_A q'$ and all computations $\lambda \in out(q', F_A)$ we have $\lambda[i] \models^c_{\text{iy}} \varphi$ for all i.
- $M, q \models^c_{\text{iy}} \langle\langle A\rangle\rangle \varphi_1 \mathcal{U} \varphi_2$ iff there exists a set of coalition-uniform y-strategies F_A such that $\forall q' \sim_A q$ and $\forall \lambda \in out(q', F_A)$, there exists a position $i \geq 0$ such that $\lambda[i] \models^c_{\text{iy}} \varphi_2$, and for all positions $0 \leq j < i$, we have $\lambda[j] \models^c_{\text{iy}} \varphi_1$.

In line with [48, 39, 49], we assume $\sim_A = \sim^D_A$ and $\approx_A = \approx^D_A$. That is, members of a coalition are able to freely communicate while executing the strategy. We believe, however, that our results carry over to the other notions of collective indistinguishability.

1 It is well known that "standard" ATL with imperfect information and perfect recall makes verification undecidable even for very simple formulae and regular models [38].

EXAMPLE 3.1: *Now, we have that* $M_1, q_{NN} \models^{c}_{ir} \langle\langle 1,2\rangle\rangle \Box \neg \mathsf{danger}$. *A successful strategy makes one guard do st in every state, and the other guard move in* $\{q_{NN}, q_{SS}\}$ *and stay elsewhere.*

4 TRANSLATING COALITION-UNIFORM ATL TO SINGLE-AGENT ATL

In this section we present a truth-preserving translation from $\text{ATL}^{c}_{\text{iy}}$ to 1ATL_{iy}.

4.1 *Reconstruction of Models*

We first propose a reconstruction of ICGS's that replaces relevant coalitions by single agents. The idea is as follows: for every coalition A occurring in a given formula φ, we remove the agents in A from Σ, and instead add a new agent a_A. The actions of a_A are combinations of actions from agents in A. Thus, the new set of agents will consist of new agents representing coalitions from φ, plus those agents that did not appear in any coalition. Now, it can be the case that some "old" agents have become part of several different "coalitional" agents a_A. If their choices agree across the new coalitional actions then the transition specified in the original model is executed. Otherwise, the system proceeds to a new "conflict" state $q_\perp$ labeled with a new atomic proposition null.

DEFINITION 4.1 (MODEL TRANSLATION): *We define a function* T *which given an* ICGS $M = \langle \Sigma, \Pi, Q, C, d, \delta, \sim, \pi \rangle$ *and an* $\text{ATL}^{c}_{\text{iy}}$ *formula* φ *with coalitions* $A_1, \ldots, A_n$ *of (where* $A_1, \ldots, A_n \subseteq \Sigma$*), translates them into a concurrent game structure* $T(M, \varphi) = M' = \langle \Sigma', \Pi', Q', C', d', \delta', \sim', \pi' \rangle$.

$\Sigma' = \{a_{A_1}, \ldots, a_{A_n}\} \cup \{\Sigma \setminus \bigcup_n A_n\} \cup \{a_d\}$ *is the new set of agents, which has new agents* $\{a_{A_1}, \ldots, a_{A_n}\}$ *that correspond to coalitions occurring in* φ*, and all the old agents except for those that belonged to coalitions in* φ*. There is also an extra agent in* Σ' *which we denote as '*a_d*' ("dummy"). For the sake of brevity we will refer to 'old' agents (those that do not belong to coalitions occurring in* φ*) as* $\{a_1, \ldots, a_m\}$*. Also, in order to be able to refer to former members of coalitions* $A_1, \ldots, A_n$*, we adopt the following notation:*

$$
\begin{aligned}
A_1 &= (a^1_1, a^1_2, \ldots, a^1_{l_1}) \\
A_2 &= (a^2_1, a^2_2, \ldots, a^2_{l_2}) \\
&\vdots \\
A_n &= (a^n_1, a^n_2, \ldots, a^n_{l_n})
\end{aligned}
$$

And we also say that there are k agents in the original structure M.

We introduce one new propositional symbol 'null' and one new state $q_\perp$*:*

$$
\begin{aligned}
\Pi' &= \Pi \cup \{null\} \\
Q' &= Q \cup \{q_\perp\}
\end{aligned}
$$

We now define the set of choices C' and a function of enabled choices d' simultaneously. We say that:

$$d'_a(q') = \begin{cases} d_a(q) & \text{if } a \in \{a_1, \ldots, a_m\} \text{ and } q' \neq q_\perp, \\ \prod_{b \in A_j} d_b(q) & \text{if } a = a_{A_j} \text{and } q' \neq q_\perp, \\ \{empty\} & \text{if } a = a_d \text{ or if } q' = q_\perp, \end{cases}$$

where $q' \in Q'$, $q \in Q$ and $a \in \Sigma'$. We say the set C' is simply an image of the function d', and we refer to members of C' as c'.

The new transition function, δ', handles non-empty intersections of coalitions that lead to (potentially) conflicting choices. Whenever two (or more) singleton coalitions have enabled choices that would lead to different states, it produces transitions to a special conflict state $q_\perp$:

$$\delta'(q', (c'_{a_1^1}, \ldots, c'_{a_{l_1}^1}), \ldots, (c'_{a_1^n}, \ldots, c'_{a_{l_n}^n}),$$

$$c'_d, c'_1, \ldots, c'_m) = \begin{cases} \delta(q, x_1, \ldots, x_k) & \text{when } A_1 \cap \ldots \cap A_n = \emptyset \text{ or} \\ & \forall_{i,j \in \{1,\ldots,n\}} \forall_{r \in \{1,\ldots,l_i\}} \forall_{s \in \{1,\ldots,l_j\}} \\ & a_r^i = a_s^j \Rightarrow c'_{a_r^i} = c'_{a_s^j} \\ q_\perp & \text{otherwise.} \end{cases}$$

where $x_i = c'_j$ if $a_i = a_j \in \{a_1, \ldots, a_m\}$ and $c'_{a_o^j}$ if $a_i = a_o^j \in A_j$.

The indistinguishability relation remains the same for old agents, and for new agents it becomes the intersection of relations for members of old coalitions:

$$\sim'_a = \begin{cases} \sim_a \cup \{q_\perp, q_\perp\} & \text{if } a \in \{a_1, \ldots, a_n\} \\ \bigcap_{b \in A_j} \sim_b & \text{if } a = a_{A_j} \\ \{q_\perp, q_\perp\} & \text{if } a = a_d \end{cases}$$

Finally, we label the special state $q_\perp$ with a new propositional symbol null: $\pi'(q) = \{null\}$ for $q = q_\perp$ and $\pi(q)$ otherwise.

EXAMPLE 4.1: *The translation of the coordinated defense model M_1 for formula $\langle\langle 1,2\rangle\rangle \Diamond \langle\langle 1,a\rangle\rangle \bigcirc$ danger is presented in Figure 4.2.*

4.2 *Translation of Formulas*

The formula translation is straightforward. We substitute each coalition A with agent a_A, and insert the proposition null so that the opponents can only lose by enforcing a conflict.

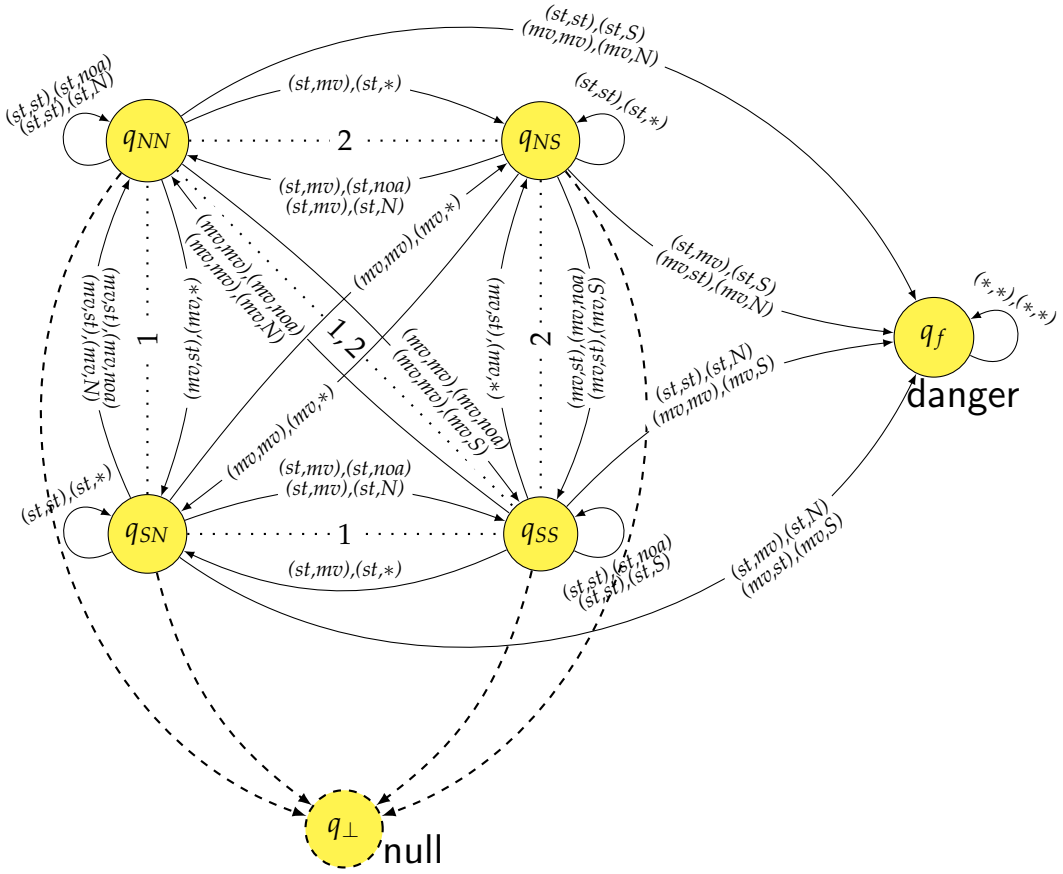

Figure 4.2: Translation of model M_1 for formula $\langle\langle 1,2\rangle\rangle\Diamond\langle\langle 1,a\rangle\rangle\bigcirc$ danger. Dashed arrows depict transitions to the "conflict" state $q_\perp$

DEFINITION 4.2 (FORMULA TRANSLATION): *We define the function* $t : \mathcal{L}_{\text{ATL}} \to \mathcal{L}_{\text{1ATL}}$ *which translates an* ATL *formula over* Σ *to a single-agent* ATL *formula over* Σ' *inductively in the following way:*

$$\begin{aligned}
t(p) &= p \\
t(\neg\varphi) &= \neg t(\varphi) \\
t(\varphi \wedge \psi) &= t(\varphi) \wedge t(\psi) \\
t(\langle\langle\emptyset\rangle\rangle\bigcirc\varphi) &= \langle\langle a_d\rangle\rangle\bigcirc(null \vee t(\varphi)) \\
t(\langle\langle\emptyset\rangle\rangle\Box\varphi) &= \langle\langle a_d\rangle\rangle\Box(null \vee t(\varphi)) \\
t(\langle\langle\emptyset\rangle\rangle\varphi\,\mathcal{U}\,\psi) &= \langle\langle a_d\rangle\rangle(null \vee t(\varphi))\,\mathcal{U}\,(null \vee t(\psi)) \\
t(\langle\langle A\rangle\rangle\bigcirc\varphi) &= \langle\langle a_A\rangle\rangle\bigcirc(null \vee t(\varphi)) \\
t(\langle\langle A\rangle\rangle\Box\varphi) &= \langle\langle a_A\rangle\rangle\Box(null \vee t(\varphi)) \\
t(\langle\langle A\rangle\rangle\varphi\,\mathcal{U}\,\psi) &= \langle\langle a_A\rangle\rangle(null \vee t(\varphi))\,\mathcal{U}\,(null \vee t(\psi))
\end{aligned}$$

where $p \in \Pi$, $null \in \Pi'$, $A \subseteq \Sigma$, *and* $a_A, a_d \in \Sigma'$.

EXAMPLE 4.2: *According to the translation, our formula* $\langle\langle 1,2\rangle\rangle\Diamond\langle\langle 1,a\rangle\rangle\bigcirc$ danger *becomes now* $\langle\langle a_{\{1,2\}}\rangle\rangle\Diamond\big(null \vee \langle\langle a_{\{1,a\}}\rangle\rangle\bigcirc(null \vee \text{danger})\big)$.

4.3 *Main Result: The Embedding Is Truth-Preserving*

In order to prove correctness of our translation we need some additional definitions and lemmas:

DEFINITION 4.3 (COMPLEXITY OF FORMULAS): *The complexity* $c : \mathcal{L}_{\mathrm{ATL}^c_{xy}} \to \mathbb{N}$ *is defined inductively as follows:*

$$\begin{aligned} c(p) &= 1 \\ c(\neg\varphi) &= 1 + c(\varphi) \\ c(\varphi \wedge \psi) &= 1 + \max(c(\varphi), c(\psi)) \\ c(\langle\langle A\rangle\rangle \bigcirc \varphi) = c(\langle\langle \emptyset\rangle\rangle \bigcirc \varphi) &= 1 + c(\varphi) \\ c(\langle\langle A\rangle\rangle \Box\varphi) = c(\langle\langle \emptyset\rangle\rangle \Box\varphi) &= 1 + c(\varphi) \\ c(\langle\langle A\rangle\rangle \varphi \mathcal{U} \psi) = c(\langle\langle \emptyset\rangle\rangle \varphi \mathcal{U} \psi) &= 1 + c(\varphi) + c(\psi) \end{aligned}$$

LEMMA 4.1: *$T(M, \neg\varphi)$ is equivalent to $T(M, \varphi)$.*

Proof. T takes a structure and a formula as its arguments, but the only part of the formula taken into consideration is the coalition predicate. Since negation does not affect it, we say that the above expressions are equivalent. □

We can now state the main result in this paper.

THEOREM 4.1: *Given an* ICGS M, *a state* $q \in M$ *and an* ATL^c_{iy} *formula* φ, *the following equivalence holds:*

$$M, q \models \varphi \text{ iff } T(M, \varphi), q \models t(\varphi)$$

Proof. Let φ be a formula. The proof is by induction on $c(\psi')$ for all subformulas ψ' of φ.

BASE CASE: If $M, q \models p$ then $T(M, \varphi), q \models t(p)$, because $t(p)$ is p, and for every $q \in Q$, $\pi'(q) = \pi(q)$. The same argument applies in the other direction, so if $T(M, \varphi), q \models t(p)$ then $M, q \models p$.

INDUCTION HYPOTHESIS: For all subformulas ψ of φ such that $c(\psi) < c(\psi')$, and all $q \in Q$: $M, q \models \psi$ iff $T(M, \varphi), q \models t(\psi)$.

CASE FOR $\psi' = \neg\psi$: Follows directly from the induction hypothesis and Lemma 4.1.

CASE FOR $\psi' = \psi_1 \wedge \psi_2$: If ψ' is a conjunction, it is trivially true that whenever $M, q \models \psi_1 \wedge \psi_2$, then $T(M, \varphi), q \models t(\psi_1 \wedge \psi_2)$. The other direction follows as well, since $t(\psi_1 \wedge \psi_2)$ translates into $t(\psi_1) \wedge t(\psi_2)$.

CASE FOR $\psi' = \langle\langle A\rangle\rangle \bigcirc \psi$:

(⇒) We want to show that:

$$M, q \models \langle\langle A\rangle\rangle \bigcirc \psi \Rightarrow T(M,\varphi), q \models \langle\langle a_A\rangle\rangle \bigcirc (null \vee t(\psi))$$

Assume that $\exists F_A \forall_{q_1 \sim_A q} \forall \lambda \in out(q_1, F_A), M, \lambda[1] \models \psi$, where $F_A = \{f_a : a \in A\}$ is a coalition-uniform set of strategies. We must show that:
$\exists f_{a_A} \forall_{q_1' \sim'_{a_A} q} \forall \lambda' \in out(q_1', f_{a_A}), T(M,\varphi), \lambda'[1] \models (null \vee t(\psi))$.
We define the strategy f_{a_A} for agent a_A in $T(M,\varphi)$ as follows: $f_{a_A}(q') = \prod_{a \in A} f_a(q')$ when $q' \in Q$, and $f_{a_A}(q') = empty$ when $q' = q_\perp$. Let $q_1' \sim'_{a_A} q$ and $\lambda' \in out(q_1', f_{a_A})$. We must show that $T(M,\varphi), \lambda'[1] \models (null \vee t(\psi))$. From Definitions 3.1 and 4.1 we have that $q_1' \sim'_{a_A} q$ implies that $q_1' \sim_A q$. That $\lambda' \in out(q_1', f_{a_A})$ means that there is a choice $c_a' \in d_a'(q_1')$ for each agent $a \in \Sigma'$ such that $c'_{a_A} = f_{a_A}(q_1')$. We now define some notation: for any $h \in \{1, \ldots, n\}$, we have that $c'_{a_{A_h}} = (c_1^h, \ldots, c_{l_h}^h) \in d'_{a_{A_h}}(q_1') = \prod_{a \in A_h} d_a(q_1')$, so for every agent $a \in \Sigma$ such that $a \in A_h = \{a_1^h, \ldots, a_{l_h}^h\}$, let $i_a^h \in \{1, \ldots, l_h\}$ be such that $a = a_{i_a^h}^h$ (a is agent number i_a^h in the enumeration of A_h). We thus have that $c_{i_a^h}^h \in d(q_1')$ is the choice of agent a in the choice $c'_{a_{A_h}}$ of the coalition A_h. We now consider two cases.

The first case is that there exist A_h and A_l such that $A_h \cap A_l \neq \emptyset$ but $c_{i_a^h}^h \neq c_{i_a^l}^l$ (the choice made by a in the two coalitions differ). In this case $\delta'(q_1', \mathbf{c}) = q_\perp$, and thus $T(M,\varphi), \lambda'[1] \models null$ and we are done.

Assume, then, the second case, that whenever there are one or more coalitions with a as a member, they all agree on the choice for agent a, i.e., $c_{i_a^h}^h = c_{i_a^l}^l$ whenever $a \in A_h \cap A_l$. We now define a choice $c_a \in d_a(q_1')$ for each agent $a \in \Sigma$ in the original model M, as follows. When $a \notin A_h$, for all h, let $c_a = c_a'$; $c_a \in d_a(q_1')$ because $d_a'(q_1') = d_a(q_1')$. When $a \in A_h$ for some h, let $c_a = c_{i_a^h}^h$ for some h such that $a \in A_h$ (this is well-defined by the assumption that all coalitions with a as a member agree on the choice of a). Let h be such that $A = A_h$ (since ψ' is a subformula of φ, A is one of the coalitions occurring in φ). Let $a \in A$. We have that $c'_{a_A} = f_{a_A}(q_1') = \prod_{a \in A} c_a$ and $c_a = c_{i_a^h}^h$ by the definitions above, and thus $c_a = c_{i_a^h}^h = f_a(q_1')$. Since $c_a = f_a(q_1')$ for any $a \in A$, there is a $\lambda \in out(q_1', F_A)$ such that $\lambda[1] = \delta(q_1', \mathbf{c})$, and we thus have that $M, \delta(q_1', \mathbf{c}) \models \psi$.[2] By the induction hypothesis, $T(M,\varphi), \delta(q_1', \mathbf{c}) \models t(\psi)$. By Definition 4.1 we have that $\delta'(q_1', \mathbf{c}') = \delta(q_1', \mathbf{c})$, and thus that $T(M,\varphi), \delta(q_1', \mathbf{c}) \models$

2 Throughout the rest of the proof we use the following notation: $\mathbf{c}$ is the action profile where the choice of agent $a \in \Sigma$ is c_a and $\mathbf{c}'$ is the action profile where the choice of agent $a \in \Sigma'$ is c_a'.

$t(\psi)$. By definition of $\mathbf{c}'$, $\delta'(q_1', \mathbf{c}') = \lambda'[1]$. Thus, $T(M,\varphi), \lambda'[1] \models t(\psi)$, and we are done.

(⇐) We want to show that:

$$T(M,\varphi), q \models \langle\langle a_A \rangle\rangle \bigcirc (null \vee t(\psi)) \Rightarrow M, q \models \langle\langle A \rangle\rangle \bigcirc \psi$$

Assume that $\exists f_{a_A} \forall_{q_1 \sim'_{a_A} q} \forall \lambda' \in out(q_1, f_{a_A}), T(M,\varphi), \lambda'[1] \models (null \vee t(\psi))$. We must show that $\exists F_A \forall_{q_1 \sim_A q} \forall \lambda \in out(q_1, F_A), M, \lambda[1] \models \psi$.

We define a coalition-uniform set of strategies $F_A = \{f_a : a \in A\}$ for coalition $A = \{a_1, \ldots, a_r\}$ as follows: for every $a = a_j \in A$ and any $q' \in Q$, $f_a(q') = c_j$, where $(c_1, \ldots, c_r) = f_{a_A}(q)$. For $q' = q_\perp$ and $a \in A$, $f_a(q') = empty$. From Definition 4.1 it is easy to see that F_A is a (collective) strategy in M (i.e., $f_a(q') \in d_a(q')$ for each $a \in A$) and from uniformity of f_{a_A} and Definition 4.1 it follows that F_A is coalition-uniform ($q' \sim_A q'' \Rightarrow f_a(q') = f_a(q'')$ for each $a \in A$).

Let $q_1 \sim_A q$. We know that $q \neq q_\perp$, because $q_\perp \notin Q$ and we also know that $q_1 \neq q_\perp$, because $q_\perp$ is indistinguishable only from itself. Hence, from Definitions 3.1 and 4.1 we get that $q_1 \sim_A q$ implies $q_1 \sim'_{a_A} q$.

Let $\lambda \in out(q_1, F_A)$. It is easy to see that also $\lambda \in out(q_1, f_{a_A})$: $T(M,\varphi)$ includes all the states of M; all the strategies F_A and f_{a_A} "do the same thing" in those states; the other agents have the same actions available in those states. Thus, $T(M,\varphi), \lambda[1] \models null \vee t(\psi)$. But it cannot be that $T(M,\varphi), \lambda[1] \models null$, because $null$ is only satisfied in $q_\perp$ and $q_\perp$ is not a state in λ (since λ is a computation in M). So, $T(M,\varphi), \lambda[1] \models t(\psi)$, and by the induction hypothesis, $M, \lambda[1] \models \psi$. Thus, $M, q \models \langle\langle A \rangle\rangle \bigcirc \psi$.

CASE FOR $\psi' = \langle\langle A \rangle\rangle \psi_1 \mathcal{U} \psi_2$:

(⇒) The proof of this case proceeds in a similar way to the previous case. We want to show that:

$$M, q \models \langle\langle A \rangle\rangle \psi_1 \mathcal{U} \psi_2 \Rightarrow T(M,\varphi), q \models \langle\langle a_A \rangle\rangle (null \vee t(\psi_1)) \mathcal{U} (null \vee t(\psi_2))$$

Assume that $\exists F_A$ such that $\forall_{q_1 \sim_A q} \forall \lambda \in out(q_1, F_A)$, there is a position $i > 0$ in λ, such that $M, \lambda[i] \models \psi_2$ and for all positions $0 \leq j < i$, $M, \lambda[j] \models \psi_1$, where $F_A = \{f_a : a \in A\}$ is a coalition-uniform set of strategies. We must show that $\exists f_{a_A} \forall_{q_1 \sim'_{a_A} q} \forall \lambda' \in out(q_1', f_{a_A})$, there is a position $i' > 0$ in λ', such that $T(M,\varphi), \lambda'[i'] \models (null \vee t(\psi_2))$ and for all positions $0 \leq j' < i'$, $T(M,\varphi), \lambda'[j'] \models (null \vee t(\psi_1))$. We define the strategy f_{a_A} for agent a_A in $T(M,\varphi)$ like before: $f_{a_A}(q') = \prod_{a \in A} f_a(q')$ when $q' \in Q$, and $f_{a_A}(q') = empty$ when $q' = q_\perp$. Let $q_1 \sim'_{a_A} q$ and $\lambda' \in out(q_1, f_{a_A})$.

It is now easy to see, in the same way as in the $\bigcirc$-case, that there exists an M-computation $\lambda \in out(q_1, F_A)$ such that either (i) $\lambda[j] = \lambda'[j]$ for all $j \geq 0$, or (ii) there exists a $k \geq 0$ such that $\lambda[j] = \lambda'[j]$ for all $0 \leq j < k$ and $\lambda'[j] = q_\perp$ for all $j \geq k$. In case (i) we are done by the induction hypothesis. We argue that we are also done in case (ii). If $k > i$ where i is such that $M, \lambda[i] \models \psi_2$, we are done by the induction hypothesis like in case (i). If $k \leq i$ we are also done: we have that $M, \lambda[j] \models \psi_1$ for all $j \leq k$; by the induction hypothesis $T(M, \varphi), \lambda[j] \models t(\psi_1)$ for all $j \leq k$ and thus $T(M, \varphi), \lambda'[j] \models t(\psi_1)$ for all $j \leq k$; and $T(M, \varphi), \lambda'[k] \models null$.

($\Leftarrow$) The proof in this direction is exactly like the proof in the same direction for the $\bigcirc$-case.

CASE FOR $\psi' = \langle\langle A \rangle\rangle \Box \psi$: analogous to the $\mathcal{U}$-case.

□

5 CONCLUSIONS

In this paper, we look closer at the issue of executable strategies for coalitions acting under imperfect or incomplete information. Based on ideas from existing literature, we propose the "coalition-uniform" semantics for Alternating-time Temporal Logic where uniformity of coalitional strategies is based on the *distributed knowledge* relation for the coalition. We also show that ATL with the new semantics can be embedded in the syntactic restriction of the logic that talks only about abilities of individual agents. This is done through a translation of models and formulae that preserves the truth of formulae in the context of a given model. We take it as a formal counterpart of our argument that coalitions whose members can fully coordinate their actions and share their knowledge should be seen as *de facto* single compound agents. We also note that the translation can be used to implement model checking of coalition-uniform ATL with verification tools for more standard variants of the logics, such as MCMAS [69] and SMC [81].

ACKNOWLEDGEMENTS. Piotr Kaźmierczak's research was supported by the Research Council of Norway project 194521 (FORMGRID). Wojciech Jamroga acknowledges the support of the National Research Fund (FNR) Luxembourg under the project GALOT (INTER/DFG/12/06), as well as the support of the 7th Framework Programme of the European Union under the Marie Curie IEF project ReVINK (PIEF-GA-2012-626398).

5

PAPER B: PLAYING WITH NORMS: TRACTABILITY OF NORMATIVE SYSTEMS FOR HOMOGENEOUS GAME STRUCTURES

This paper was presented during the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014) on May 9th, 2014 in Paris, France. It has been published in the conference proceedings.

PLAYING WITH NORMS: TRACTABILITY OF NORMATIVE SYSTEMS FOR HOMOGENEOUS GAME STRUCTURES

Sjur Dyrkolbotn
Durham School of Law
Durham University, UK

Piotr Kaźmierczak
Dept. of Computing, Mathematics and Physics
Bergen University College, Norway

Abstract

In this paper we study the connection between anonymous (normal form) games and normative systems for homogeneous concurrent game structures. We present a variant of the popular strategic logic ATL that allows for reasoning about norm compliance and the strategic ability of coalitions of agents. We broaden the notion of a normative system compared to earlier work and introduce two categories: individual and collective norms. Then we establish a technical link between these two notions, and between the notion of an individual norm and an anonymous game, as studied in recent work on algorithmic game theory. These connections enable us to show that model checking on homogeneous concurrent game structures with norms is tractable in the number of agents.

1 INTRODUCTION

Normative systems (or *social laws*) have received a lot of attention in the multi-agent systems community in recent years, and have proved to be a very useful framework for agent coordination.[1] The idea is simple: we put *behavioral constraints* on agents and check whether they can achieve given *objectives* while complying to these constraints. While initially proposed and analyzed by Shoham & Tennenholtz [95, 96], normative systems were further studied using modal logic by Ågotnes *et al.* [4, 5, 7, 8] with *Computation Tree Logic* (CTL) as a language for expressing objectives and agent-labeled Kripke structures as system models, and by van der Hoek *et al.* [102] who used *Alternating-time temporal logic* (ATL) to express objective and *Concurrent Game Structures* (CGSS) as models.

1 Throughout the paper we use 'normative systems' or 'norms' whenever we refer to behavioral constraints on our agents. It is similar to the notion of a 'social law' as defined by Shoham & Tennenholtz [95, 96].

The normative systems used in these papers suffer from two major shortcomings:

1. They are not expressive enough: A normative system is understood as a list of forbidden actions, one list per agent, per state of the system. That does not allow us to model *normative interaction*. There is no coordination involved in norm compliance – the normative constraints imposed on an agent's behavior do not in any way depend on the actions performed by the other agents.

2. They are not tractable: Checking whether or not a given normative system can ensure that a given objective is met is typically (co)-NP-complete, and exact algorithms tend to use time exponential in the number of agents.

In this paper we address both of these shortcomings by introducing a logic for reasoning about a broader notion of norms, for which we can also show that model checking can be done in time polynomial in the number of agents in the system. This means that the logic is tractable for analyzing and verifying properties of large scale systems containing a great number of agents, as long as it involves a constant number of other elements, such as states and actions.

Tractability follows from imposing a restriction on the underlying system, formulated with respect to CGS's. We require, in particular, that the system is *homogeneous*, meaning that it allows a branching-time future that depends solely on the *number* of agents performing various actions, and does not differentiate based on agents' identity. This restriction stems from [80] where an equivalent role-based semantics for ATL was given to facilitate fast model checking of models with few roles. Homogeneity arises from restricting attention to models with a single role, and the corresponding class of structures was axiomatized in [78].

The norms we consider involve no corresponding homogeneity restriction and they can prescribe different sets of legal actions to different agents, depending on the state of the system. Furthermore, we introduce norms that allow us to model coordination, such that the set of legal actions for an agent depends on the actions performed by other agents, up to equivalence with respect to the underlying structure. We consider two ways of doing this:

- Individually: Each agent has a list of forbidden actions, but the list depends on what the other agents do.

- Collectively: The normative system forbids tuples of actions directly.

We demonstrate that these two notions are equivalent as long as we assume full norm compliance. Then we discuss partial compliance and argue that individual norms raise some conceptual questions that we can only answer successfully if we recognize that individual and collective

norms are genuinely different, and require different notions of partial compliance in order to do justice to fundamental intuitions and modeling aims.

Adding normative systems to homogeneous structures results in a logic that no longer exhibits homogeneity with respect to the strategic ability of agents under norm compliance. Hence norms serve to *reintroduce* individuality to the system. However, we show that model checking remains tractable. This follows from a link with recent work in algorithmic game theory, namely showing how normative systems can be efficiently *incentivized* by a normal form game which rewards agents for following the norm. We observe, in particular, that the set of Nash Equilibria for this game corresponds exactly to the legal actions for the agents under the corresponding normative system. Furthermore, we show that all such normative games belong to a class of games which admit compact representation facilitating polynomial time computation of the set of Nash Equilibria.

In Section 2 we introduce basic notation and present a terse summary of the main technical concepts we rely on. Then, in Section 3 we introduce formal definitions of collective and individual norms for homogeneous concurrent game structures and present a technical elaboration of these concepts that culminates in the definition of truth for a language of norm compliance on homogeneous structures with norms. In Section 4 we present our main result, encoding normative systems as anonymous games, and we use this correspondence to establish tractability of model checking. We conclude and discuss directions for future work in Section 5.

2 PRELIMINARIES

The technical results presented in this paper arise from a novel combination of results on three distinct formal notions: homogeneous concurrent game structures, anonymous normal form games, and normative systems. We now proceed with detailed, but somewhat terse, definitions of all the necessary formal background regarding the first two concepts, and we devote a separate section to normative systems, since our definitions significantly broaden the notion of a norm previously considered in related work on strategic logics for multi-agent systems.

2.1 *Notation and basic concepts*

We start by introducing some notation and basic definitions that we will use in the remainder of the paper. Throughout we assume given a set $\Sigma = \{1, \ldots, n\}$ of agents and a set $\mathbb{A} = \{p_1, \ldots, p_m\}$ of actions.[2] We say that given any function $s : X \to Y$ we let $dom(s) = X$ denote its domain and $targ(s) = \{y \in Y \mid \exists x \in X : s(x) = y\}$ denote the subset of the target

2 We will only consider semantic structures for which the actions are shared among the agents.

that s is onto. We let s_{-i} denote the function $s \restriction dom(s) \setminus \{i\}$ (where $\restriction$ is used to signify partial function application) and we let (s_{-i}, p) denote the function s' defined by:

$$s'(j) = \begin{cases} s(j) \text{ if } j \neq i \\ p \text{ otherwise.} \end{cases}$$

Also, for an arbitrary function $s : X \to Y$ we let $s^- : Y \to 2^X$ denote the function defined by:

$$s^-(y) = \{x \mid s(x) = y\}$$

for all $y \in Y$. We will often use the notation $\#(y, s) = |s^-(y)| = |\{x \in X \mid s(x) = y\}|$ and $(\#(y, s))_{y \in Y}$, which we will refer to as the *profile induced by* s. It is a vector which records, for each $y \in Y$, the number of elements of X that "choose" y. Given a coalition $C \subseteq \Sigma$, an action-tuple for C is a function $s \subseteq \mathbb{A}^C$, where $s(i)$ is player i's action. Hence the function-space $\mathbb{A}^C$ contains all possible action-tuples for the coalition C. Using this notation, we define the set $P^C(\mathbb{A}) = \{(\#(y, s))_{y \in \mathbb{A}} \mid s \in \mathbb{A}^C\}$. We refer to elements $F \in P^C(\mathbb{A})$ as *C-profiles* or just partial profiles if C is not specified. Given a profile $F \in P^C(\mathbb{A})$ and an action $p_1 \in \mathbb{A}$, we use (F, p_1) to denote the profile F' defined by

$$F'(p_2) = \begin{cases} F(p_2) + 1 \text{ if } p_2 = p_1 \\ F(p_2) \text{ otherwise.} \end{cases}$$

We let F_{-p_1} denote the profile F' defined by

$$F'(p_2) = \begin{cases} F(p_1) - 1 \text{ if } p_2 = p_1 \\ F(p_2) \text{ otherwise.} \end{cases}$$

Moreover, for two profiles F, F' the profile $F + F'$ is defined for all $p \in \mathbb{A}$ as follows:

$$F + F'(p) = F(p) + F'(p)$$

For all $F \in P^C(\mathbb{A})$, we also use $ext(F) = \{F' \in P^\Sigma(\mathbb{A}) \mid F' \restriction C = F\}$. We notice that we can generate the set $P^C(\mathbb{A})$ without making use of the set of action-profiles for C, hence we avoid traversing a set that has exponential size in the number of agents. In particular, given a coalition C and a vector $F \in \mathbb{N}^\mathbb{A}$ such that $\sum_{p \in \mathbb{A}} F(p) = |C|$, we observe that there is at least one action-tuple $s_F \in \mathbb{A}^C$ such that $(\#(y, s_F))_{y \in \mathbb{A}} = F$. Conversely, we have, for all $s \in \mathbb{A}^C$, $\sum_{p \in \mathbb{A}} \#(p, s) = |C|$. It follows that $P^C(\mathbb{A}) = \{F \in \mathbb{N}^\mathbb{A} \mid \sum_{p \in \mathbb{A}} F(p) = |C|\}$ is a compact characterization of the set of profiles for a coalition.

2.2 *NCHATL and Concurrent Game Structures*

The logical language we use, $\mathcal{L}_{\text{NCHATL}}$, is based on ATL [12], extended with one extra operator that we use to express norm compliance. Formally, the language is generated by the following BNF:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\langle C\rangle\rangle \bigcirc \varphi \mid \langle\langle C\rangle\rangle \Box\varphi \mid \langle\langle C\rangle\rangle \varphi \mathcal{U} \varphi \mid \langle C\rangle \varphi$$

where p is a propositional symbol, and $C \subseteq \Sigma$ is a coalition of agents.[3]

The language of NCHATL contains three types of modalities for talking about pairs (H, χ) where H is a semantic structure and χ is a normative system.

- $\bigcirc$, $\Box$ and $\mathcal{U}$ are standard *temporal* operators known from many temporal logics, and stand for "next state", "some future state" and "until", respectively;
- $\langle\langle C\rangle\rangle$ is a *strategic ability* operator, and its intuitive meaning is that the coalition $\langle\langle C\rangle\rangle \bigcirc \varphi$ has a *joint strategy* for enforcing a formula φ in the next state of H;
- finally $\langle C\rangle$ is the *norm compliance* operator, which has the intuitive reading that the agents in C are willing to comply to χ.

For example, if we have some objective φ and we wonder if C complying to χ enables D to ensure φ in the next state of H, we would test if the formula $\langle C\rangle\langle\langle D\rangle\rangle \bigcirc \varphi$ is true on (H, χ). In this way, properties of H that depend on the choices of the agents can be specified with great flexibility using χ, without altering the specification of the underlying system H. In particular, rather than hard-coding predictions and intentions concerning agent behavior in the system design, we give this aspect a separate semantic characterization using χ, enabling us also to address and study interactions between the distinct modalities of norm compliance and strategic ability. Further motivation for using normative systems in verification and specification of multi-agent systems can be found in [8].

In the following, we specify H and χ compactly, and in a way that allows for great freedom in the choice of χ. To this end, we first recap the definition of a *concurrent game structure* [12], which provides the backbone for our definition of H.

DEFINITION 2.1: *A* CGS *is a tuple* $S = \langle \Sigma, Q, \Pi, (\mathbb{A}_q)_{q\in Q}, \pi, \delta\rangle$ *where:*

- Q *is the non-empty set of states.*
- Π *is a set of propositional letters and* $\pi : Q \to 2^{\Pi}$ *maps each state to the set of propositions true in it.*
- $\mathbb{A}_q \subseteq \mathbb{A}$ *is the set of actions available at* q.

3 This language closely resembles the language of *Norm Compliance* CTL presented in [8].

- *δ is the transition function. For each $q \in Q$ and any action-tuple $s \in \mathbb{A}_q^\Sigma$, it returns a state $q' = \delta(q, s) \in Q$, referred to as a* successor *of q.*

The crucial condition which allows for tractable model checking of $\mathcal{L}_{\text{NCHATL}}$ is *homogeneity*, which was first introduced (implicitly) in [80], and subsequently axiomatized in [78]. It is formalized as follows, such that S is said to be homogeneous if the following hold, for all $q \in Q$ and all action-tuples $s, s' \in \mathbb{A}_q^\Sigma$:

$$(\#(y, s))_{y \in \mathbb{A}_q} = (\#(y, s'))_{y \in \mathbb{A}_q} \Rightarrow \delta(q, s) = \delta(q, s') \tag{5.1}$$

Alternatively, the axiomatic characterization in terms of ATL (using the language of $\mathcal{L}_{\text{NCHATL}}$ without the norm-compliance operator) is also very intuitive, saying that a CGS is homogeneous as long as any two coalitions of the same size have the same strategic ability in the next-time step:

$$\langle\langle C \rangle\rangle \bigcirc \varphi \leftrightarrow \langle\langle D \rangle\rangle \bigcirc \varphi \text{ if } |C| = |D| \tag{5.2}$$

For space reasons we omit the formal definition of truth on CGSs, and only remark that the crucial property of homogeneous CGSs is that they permit testing the truth of ATL-formulas with only a polynomial time dependence on the number of agents.[4] This result follows from compact representation of such structures, obtained by replacing explicit action-tuples by profiles. In particular, a homogeneous CGS can be equivalently represented by a structure of the following kind [80, 78]:

DEFINITION 2.2: *An* HCGS *is a tuple $H = \langle \Sigma, Q, \Pi, \pi, (\mathbb{A}_q)_{q \in Q}, \delta \rangle$ where*

- *Σ, Q, Π, π and $(\mathbb{A}_q)_{q \in Q}$ are defined as in Definition 2.1,*
- *For every state $q \in Q$, and every $F \in P(q) = P^\Sigma(\mathbb{A}_q)$, δ maps F to a successor, $\delta(q, F) = q' \in Q$.*

The profiles $F \in P(q)$ assign a natural number to each action such that the sum of these numbers (over all actions) sums up to n (the number of agents). The intended meaning is that the profile describes, for all actions, how many agents perform that action. We also have C-profiles at $q \in Q$, for all $q \in Q, C \subseteq \Sigma$

$$P^C(q) = P^C(\mathbb{A}_q) = \left\{ F \in \mathbb{N}^{\mathbb{A}_q} \;\middle|\; \sum_{p \in \mathbb{A}_q} F(p) = |C| \right\} \tag{5.3}$$

EXAMPLE 2.1: *Consider an example of an* HCGS *that models a jazz trio.*

- *The trio consists of three players: Keith, Gary and Jack. $\Sigma = \{k, g, j\}$.*

4 This is not true for CGSs in general since their size is typically exponential in the number of agents. There are, in particular, exponentially many distinct action-tuples at every state. This is a major argument against the usefulness of ATL in practical multi-agent modeling scenarios, as pointed out, for instance, in [57].

- *A scenario that we are modeling is a concert, which makes for two states: one where the players are idle (pre-performance), and one in which they are playing.* $Q = \{q_0, q_1\}$, $\Pi = \{idle, play\}$, $\pi(q_0) = \{idle\}, \pi(q_1) = \{play\}$.

- *Agents have four actions available in the initial state: to remain idle, to play the piano, to play the double-bass, and to play the drums, which are actions* p_1, p_2, p_3 *and* p_4, *respectively. In state* q_1, *agents can perform action done, which is action* p_5. $\mathbb{A}_{q_0} = \{p_1, p_2, p_3, p_4\}$, $\mathbb{A}_{q_1} = \{p_5\}$.

- *Every agent is allowed to play, and to become 'idle' once the concert is over. We only list transitions that lead from one state to a new one, all other possible transitions are reflexive arrows. In particular, we have* $\delta(q_0, 0, 1, 1, 1) = q_1$ *and* $\delta(q_1, 3) = q_0$.

We return to this scenario in Example 3.1 when we introduce a normative system to help ensure that the concert becomes a success.

We postpone a formal definition of truth on HCGSs until Definition 3.4, when we will define truth for (H, χ), where H is an HCGS and χ is a normative system. Before we turn to a formalization of norms, we present the second basic building block of our approach, namely *anonymity* in the sense of algorithmic game theory [20, 22].

2.3 *Anonymous games*

A normative system describes constraints on behavior, but does not provide any mechanism for making sure that agents conform to these constraints. To do this, one may associate a mechanism of reward and/or punishment with the normative system, to provide an incentive for agents to follow the norm. We will do this in Section 4, partly to show how our norms can be backed up by an incentive mechanism, but also to show how recent work in algorithmic game theory can be used to ensure efficient model checking. In the following we provide the necessary background on the game theoretic notions we rely on.

DEFINITION 2.3: *A (normal form)* game *(with shared actions) is a tuple* $\Gamma = (\Sigma, \mathbb{A}, (\mu_i)_{i \in \Sigma})$ *such that for all action-tuples* $s \in \mathbb{A}^\Sigma$, $\mu_i(s) \in \mathbb{R}$ *is the* payoff *for agent* i. *We also define the following notions:*

- *For any action-tuple* s, *agent* i*'s best response to* s *is* $\mathsf{b}_i(s) = \{p \in \mathbb{A} \mid \forall p' \in \mathbb{A} : \mu_i(s_{-i}, p) \geq \mu_i(s_{-i}, p')\}$.

- *An action-tuple* s *is a Nash Equilibrium in pure strategies if for all* $i \in \Sigma$ *we have* $s(i) \in \mathsf{b}_i(s)$. *That is, no player can do better by changing his action, assuming all other actions remain fixed. We let* $\mathsf{NE}(\Gamma)$ *denote the set of all Nash Equilibria in* Γ.

We notice that for any game with shared actions the number of distinct Σ-profiles is polynomial in Σ. This motivates studying classes of games for which the payoffs of players remain invariant under (classes of) action-tuples that induce the same profile. The following two notions from recent work in game-theory both rely on this idea [20, 22].

DEFINITION 2.4: *A game is said to be:*

- Homogeneous *if for all agents* $i \in \Sigma$ *and all action-tuples* $s, s' \in \mathbb{A}^\Sigma$, *we have*
$$(\#(y,s))_{y\in\mathbb{A}} = (\#(y,s'))_{y\in\mathbb{A}} \Rightarrow \mu_i(s) = \mu_i(s')$$
- Anonymous *if for all agents* $i \in \Sigma$ *and all action tuples* $s, s' \in \mathbb{A}^\Sigma$, *we have*
$$s(i) = s'(i) \;\&\; ((\#(y,s_{-i}))_{y\in\mathbb{A}} = (\#(y,s'_{-i}))_{y\in\mathbb{A}} \Rightarrow \mu_i(s) = \mu_i(s')$$

In short, the intuition behind homogeneous games is that outcomes for the players remain invariant under permutations of actions over the players.[5] In anonymous games, agents may differentiate between different tuples that induce the same profile, but only with respect to their own action; they do not care about the identity of the agents performing the remaining actions that induces the profile.

For homogeneous and anonymous games, we can express the payoff for each player as a function of profiles rather than explicit action-tuples. To this end we first define, for all $C \subseteq \Sigma, F \in P^C(\mathbb{A})$ and all $1 \leq k \leq m$ the following number, which will allow us to define a canonical action-tuple s_F that induces F for any $F \in P^C(\mathbb{A})$. Keep in mind here that $\mathbb{A} = \{p_1, \ldots, p_m\}$, since we will rely on this ordering below.

$$\begin{aligned} &\rho(0,F) = 1 \;\&\; \rho(1,F) = F(p_1) \\ &\rho(i,F) = \rho(i-1,F) + F(p_i) \text{ for all } 2 \leq i \leq m \end{aligned} \tag{5.4}$$

Then we can define $s_F \in \mathbb{A}^C$ as follows, for all $F \in P^C(\mathbb{A}), i \in \Sigma$:

$$s_F(i) = p_j \text{ for } j \text{ such that } \rho(j-1,F) \leq i \leq \rho(j,F) \tag{5.5}$$

Clearly, we have $(\#(y,s_F))_{y\in\mathbb{A}} = F$, and this representation (easily computable in polynomial time) can now be used to define a compact version of the payoff function for homogeneous and anonymous games.

$$\mu_i(F,p) ::= \mu_i(s_F,p) \text{ for all } F \in P^{\Sigma\setminus\{i\}}(\mathbb{A}), p \in \mathbb{A}$$

The action p in $\mu_i(F,p)$ is to be thought of as the action performed by agent i. Notice that we need to keep track of this p explicitly in order

5 This is sometimes called *self-anonymous* in the game theory literature.

to also cover the case of anonymous games. We may now lift the best response function accordingly, for all agents $i \in \Sigma$.

$$\forall F \in P^{\Sigma\setminus\{i\}}(\mathbb{A}): \quad \mathsf{b}_i(F) = \{p \mid \forall r \in \mathbb{A} : \mu_i(F,r) \leq \mu_i(F,p)\} \tag{5.6}$$

From the definition of anonymity and homogeneity, it immediately follows that this representation is faithful, in the following sense, for all $s \in \mathbb{A}^\Sigma$:

$$\mathsf{b}_i(s) = \mathsf{b}_i((\#(y, s_{-i}))_{y\in\mathbb{A}}) \tag{5.7}$$

The crucial property of homogeneous and anonymous games is that the set of Nash Equilibria can be computed by looking only at the representation that relies on profiles rather than explicit action-tuples. In particular, as first observed in [20] and elaborated upon in [22], it follows from Hall's marriage theorem, a fundamental result from combinatorics [50], that a profile $F \in P^\Sigma(\mathbb{A})$ corresponds to a Nash Equilibrium s of Γ with $(\#(y,s))_{y\in\mathbb{A}} = F$ if, and only if, the following condition holds:

$$\forall P \subseteq \mathbb{A} : |\{i \in \Sigma \mid \exists p \in P : p \in \mathsf{b}_i(s_F)\}| \geq \sum_{p\in P} F(p) \tag{5.8}$$

From this characterization and Equation 5.7 it is easy to see that computing the set of profiles corresponding to $\mathsf{NE}(\Gamma)$ can be done in polynomial time in the number of agents. In fact, as shown in [22], the problem is in the complexity class TC_0. We will make use of this fact later to show that the normative systems for homogeneous structures admit tractable model checking procedures.

3 NORMATIVE SYSTEMS AND TRUTH

Unlike previous work on formal logical representation of normative systems, we do not restrict attention to simple lists of forbidden actions. Rather, we want to consider norms that reflect the way in which norms are inherently *social*, such that what an agent should do can depend on what other agents are doing. Compliance to such norms might then require coordination – they can be impossible to comply to unless the agents jointly reach a decision about what to do, so that each agent knows what actions the other agents intend to perform. In light of this, one may also consider norms that do not consist in specifying illegal actions for individual agents, but rather specifies directly the joint actions that are not allowed. In particular, one may consider both *individual* and *collective* normative systems.

Below we provide definitions of both these kinds of normative systems for homogeneous concurrent game structures. Both of them potentially involve coordination, and we observe that under the assumption of full norm compliance the two notions are the equivalent. We argue, however,

that for partial compliance, a non-obvious notion is needed to do justice to individual norms, setting them apart from collective ones. The basic definition is as follows:

DEFINITION 3.1: *Given an* HCGS H,

- *An individual norm is a collection of functions* $\eta = \{\eta_i \mid i \in \Sigma\}$, *such that for all* $q \in Q, i \in \Sigma, F \in P^{\Sigma\setminus\{i\}}(q)$ *we have*

$$\eta_i(q, F) \subset \mathbb{A}_q$$

- *A collective norm is function* κ *such that for all* $q \in Q$

$$\kappa(q) \subset P(q)$$

The normative systems considered in previous work on strategic logics correspond to a restricted class of individual norms, defined below.

DEFINITION 3.2: *Given an* HCGS H *and an individual normative system* η, *we say that* η *is* simple *if for all* $q \in Q, i \in \Sigma$ *and all* $F, F' \in P^{\Sigma\setminus\{i\}}(q)$

$$\eta_i(q, F) = \eta_i(q, F')$$

Simple norms capture normative systems/social laws for which the forbidden actions only depend on the state, as in [8, 102]. Given an individual norm η we define:

$$\mathsf{L}_\eta(q) = \{s \in \mathbb{A}^\Sigma \mid \forall i \in \Sigma : s(i) \in \mathbb{A}_q \setminus \eta_i(q, (\#(y, s_{-i}))_{y \in \mathbb{A}_q})\} \tag{5.9}$$

These are the *legal* action-tuples at q, when we assume full compliance to the normative system η; everyone chooses an action that is permitted in light of what all the other agents choose to do. We say that an individual normative system is *consistent* if for all $q \in Q$ we have $\mathsf{L}_\eta(q) \neq \emptyset$. Intuitively, consistency of a normative system means that it is possible to comply with it, and in the following we assume that all normative systems are consistent.[6]

We also define the legal action tuples at q when we assume full compliance to a collective norm κ:

$$\mathsf{L}_\kappa(q) = \{s \in \mathbb{A}^\Sigma \mid (\#(y, s))_{y \in \mathbb{A}_q} \in P(q) \setminus \kappa(q)\} \tag{5.10}$$

Let us now define:

$$P_\eta(q) = \{(\#(y, s))_{y \in \mathbb{A}_q} \mid s \in \mathsf{L}_\eta(q)\} \tag{5.11}$$

6 It will follow from Theorem 4.1 that consistency can be checked in polynomial time. The requirement is made here to ensure that our models remain serial after implementation of a norm. Notice that all simple norms are consistent, since all agents have at least one legal action at every state. We mention that this is a standard assumption from the literature, see e.g., [8].

These are the profiles induced by the legal action tuples at q, and they provide us with a translation of individual norms into collective norms. In particular, given an individual norm η we obtain the collective norm ${}_{\eta}\kappa$ defined at all $q \in Q$ by

$$ {}_{\eta}\kappa(q) = P(q) \setminus P_{\eta}(q) \tag{5.12} $$

Let us now go the other way; from collective norms to individual norms. Assume we have given a collective norm κ. Then we first choose some profiles $f = \{f_q \in \mathbb{N}^{\mathbb{A}} \mid q \in Q\}$ such that for all $q \in Q$ we have $f_q \in P(q) \setminus \kappa(q)$. This allows us to represent the norm κ by the individual norm ${}^{f}_{\kappa}\eta$ defined as follows, for all $q \in Q, i \in \Sigma$ and $F \in P^{\Sigma\setminus\{i\}}(q)$:

$$ {}^{f}_{\kappa}\eta_i(q, F) = \{p \in \mathbb{A}_q \mid p \neq s_{f_q}(i) \text{ and } (F, p) \in \kappa(q)\} \tag{5.13} $$

Here s_{f_q} is the canonical action-tuple inducing f_q, as defined in Equation 5.5. Adequacy of the translation is now easy to establish, and we omit the proof for space reasons.[7]

PROPOSITION 3.1: *For any* HCGS H *and any normative system* κ *on* H*, we have* $P_{{}^{f}_{\kappa}\eta}(q) = \kappa(q)$ *for all* $q \in Q$.

We now turn to partial compliance, first for collective norms. Given a coalition C, we define the set of C-profiles that are compliant to κ as follows:

$$ P^{C}_{\kappa}(q) = \{F \in P^{C}(q) \mid ext(F) \cap (P(q) \setminus \kappa(q)) \neq \emptyset\} \tag{5.14} $$

So a C-profile is compliant to κ if it can be extended to a complete profile that is legal.[8]

Let us now turn to partial compliance for individual norms. It is tempting to say that partial compliance by $C \subseteq \Sigma$ to η can be defined using the translation to collective norms directly, as follows:

$$ \{F \in P^{C}(q) \mid ext(F) \cap P_{\eta}(q) \neq \emptyset\} \tag{5.15} $$

However, this is not adequate since it admits F as a partially compliant C-profile whenever F can be induced by a choice of actions for *some* coalition of size $|C|$ that adhere to η. It does not ensure that F can be induced by a choice of actions that is permissible for C. Hence for individual norms, the notion defined below is more appropriate. It refers to the set of explicit action-tuples to ensure that a partially compliant profile for C can actually be induced by actions that are allowed for C. In particular, we define the

7 In particular, it is straightforward to see that we allow all profiles that are permitted under κ. However, to ensure that every agent always has some legal action to perform, we allow by default the canonical actions that induces the legal profiles f_q, for all $q \in Q$.

8 For future work we would like to consider more subtle notions of compliance, but this seems like the obvious place to start, a minimum requirement that is hard to dispute.

set of C-profiles that are partially compliant to an individual norm η as follows, for all $q \in Q$:

$$P_\eta^C(q) = \{(\#(y,s))_{y\in\mathbb{A}_q} \mid s \in \mathbb{A}^C \;\&\; \exists s' \in \mathsf{L}_\eta(q) : s = s' \restriction C\} \tag{5.16}$$

Hence we require that the C-profile can be induced by a C-action-tuple which can be extended to a legal action-tuple. This clearly implies that the profile can be extended to a legal profile, but it is stronger; it also requires the agents in C to act in such a way as *they* are permitted to act under η. It is not enough that *some* collection of $|C|$ agents could have acted in this way.

3.1 *Truth for* NCHATL *on* (H, χ)

We are almost ready to define truth of $\mathcal{L}_{\text{NCHATL}}$ formulas on (H, χ) where H is an HCGS and χ is an individual or a collective norm. In order to do so, we first need to define the notion of a *compliant strategy* with respect to a norm $\chi \in \{\eta, \kappa\}$ which is either collective or individual. This, in turn, requires us to define the set of legal D-profiles given C-compliance to χ:

$$\begin{aligned} &P_\chi^C(q, D) = \{F \in P^D(q) \mid \\ &\exists F_1 \in P_\chi^{C\cap D}(q) : \exists F_2 \in P^{D\setminus C}(q) : F = F_1 + F_2\} \end{aligned} \tag{5.17}$$

Notice that this definition requires $C \cap D$ to comply with the norm *independently* from the remaining agents in C. For collective and simple norms, this makes no difference, but for individual norms involving coordination this means that we require something stronger than norm-compliance of C as a coalition. Rather, we interpret D as the coalition of agents which coordinate their actions, so the fact that C follow the norm is taken to mean that when D is acting as a group, without necessarily including all members of C, then $C \cap D$ also follow the norm as a sub-group of C. However, we also require, when defining the extensions of D's strategy below, that C follow the norm as a whole. Hence in effect we model the situation when $C \cap D$ and $C \cap (\Sigma \setminus D)$ follow the norm independently of one another.

We could define this differently by assuming only that C follow the norm as a whole and do not necessarily coordinate their actions inside D to meet the requirements of η. Nothing hinges on this for the results that are about to follow, but our solution seems more natural with respect to the intended readings of $\langle\langle D\rangle\rangle$ and $\langle C\rangle$.

We say that a *computation* is an infinite sequence $\lambda = q_0 q_1 \ldots$ of states such that for all positions $i \geq 0$, q_{i+1} is a successor of q_i and we follow standard abbreviations. Hence a q-computation denotes a computation starting at q, and $\lambda[i]$, $\lambda[0,i]$ and $\lambda[i,\infty]$ denote the i-th state, the finite prefix $q_0 q_1 \ldots q_i$ and the infinite suffix $q_i q_{i+1} \ldots$ of λ for any computation λ and its position $i \geq 0$, respectively.

This leads us to the following notion of a *strategy* for a coalition D assuming that C complies with the normative system.

DEFINITION 3.3: *A χ $\restriction$ C-compatible D-strategy is a map $s_D : Q \to \bigcup_{q \in Q} P^C_\chi(q, D)$ such that*

$$s_D(q) \in P^C_\chi(q, D) \text{ for each } q \in Q$$

We denote the set of all such strategies by $strat^\chi_C(D)$.

Notice that if $s \in strat^\chi_C(\Sigma)$ for some $C \subseteq \Sigma$, then if we apply $\delta(q)$ to $s(q)$ we obtain a unique new state $q' = \delta(q, s(q))$. Iterating, we get the *induced* computation $\lambda_{s,q} = q_0 q_1 \ldots$ such that $q = q_0$ and $\forall i \geq 0 : \delta(q_i, (s(q_i))) = q_{i+1}$. Given $s_D \in strat^\chi_C(D)$ and a state q we get an associated *set* of computations $out(s_D, q)$. This is the set of all computations that can result when at any state, D is acting in the way specified by s_D. That is

$$out(s_D, q) := \{\lambda_{s,q} \mid s \in strat^\chi_C(\Sigma) \text{ and } s_D \leq s\} \tag{5.18}$$

We are now ready for the main definition of this section.[9]

DEFINITION 3.4: *Given a normative* HCGS (H, χ) *with $\chi \in \{\eta, \kappa\}$ being either a collective or individual norm, a state q and a coalition $C \subseteq \Sigma$, truth of φ on (H, χ) under C-compliance is defined inductively.*

- $H, \chi, C, q \models p$ *iff* $q \in \pi(p)$
- $H, \chi, C, q \models \neg\varphi$ *iff* $H, \chi, C, q \not\models \varphi$
- $H, \chi, C, q \models \varphi \vee \psi$ *iff* $H, \chi, C, q \models \varphi$ *or* $H, \chi, C, q \models \psi$
- $H, \chi, C, q \models \langle\langle D \rangle\rangle \bigcirc \varphi$ *iff* $\exists s_D \in strat^\chi_C(D) : \forall \lambda \in out(s_D, q) : \lambda[1] \models \varphi$
- $H, \chi, C, q \models \langle\langle D \rangle\rangle \Box \varphi$ *iff* $\exists s_D \in strat^\chi_C(D) : \forall \lambda \in out(s_D, q) : \forall i \geq 0 : \lambda[i] \models \varphi$
- $H, \chi, C, q \models \langle\langle D \rangle\rangle \varphi \mathcal{U} \psi$ *iff* $\exists s_D \in strat^\chi_C(D) : \forall \lambda \in out(s_D, q) : \exists i \geq 0 : (\lambda[i] \models \psi \wedge \forall j \in [i] : \lambda[j] \models \varphi)$
- $H, \chi, C, q \models \langle D \rangle \varphi$ *iff* $H, \chi, D, q \models \varphi$

EXAMPLE 3.1: *We continue our running jazz trio example. The trio has one objective – to play the concert – which we can express by the following* NCHATL *formula: $\langle\langle \{k, g, j\} \rangle\rangle \bigcirc play$. This formula is true on the model, but it does not* guarantee *a concert. For that we need to ensure this one: $[\![\{k, g, j\}]\!] \bigcirc play$. We can design a set of individual norms to make this formula true, assuming that all agents comply:*

- $\eta_k(q_0, \langle x, y, z, w \rangle) = \eta_g(q_0, \langle x, y, z, w \rangle) = \eta_j(q_0, \langle x, y, z, w \rangle) = \{p_2, p_3, p_4\}$ *if* $x \neq 0$
- $\eta_k(q_0, \langle 0, 0, 1, 1 \rangle) = \{p_1, p_3, p_4\}$;

9 Notice that our definition requires also C to follow the norm as a group, when we compute the set of possible extensions of D's strategy.

- $\eta_g(q_0, \langle 0,1,0,1\rangle) = \{p_1, p_2, p_4\}$;
- $\eta_j(q_0, \langle 0,1,1,0\rangle) = \{p_1, p_2, p_3\}$;

The first point ensures that an agent is allowed to play only *when all the other agents also play an instrument. So if we add a new state to the model, for the case that only some of the agents play, we would now be able to prevent this state from coming about by implementing this normative system. This illustrates the increased expressive power we get from considering coordination. Also note that under full compliance we are sure that the agents play the instruments that they* know *how to play: Keith plays piano, Gary plays the double-bass and Jack plays the drums.*

But at this point we also see the limit of our approach, since with an underlying homogeneous structure, we cannot distinguish this from a situation where everyone plays but plays a different instrument than his own. For instance, even if Jack and Gary switch their instruments, our norm still requires Keith to play, even if the concert should probably not go ahead in this case. However, we can capture (parts of) this distinction, since it corresponds to a scenario when the norm is violated by Jack and Gary. A direction for future research that we think will be fruitful is to attempt to use the norm structure more actively, by developing extensions of the logic for describing how bringing about a concert in the wrong way, in violation of the norm, could influence the future development of the system. This could perhaps be modeled by letting the violation itself directly influence the agents' beliefs, desires and intentions. A concert is a concert, one might say, but the audience might not want to come to the next one if Jack the drummer suddenly starts playing the double-bass.

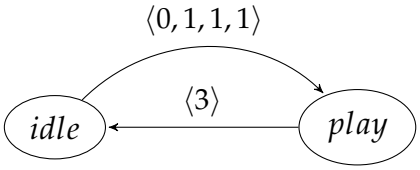

Figure 5.1: Jazz trio example (reflexive arrows omitted).

4 PARTIAL COMPLIANCE IS TRACTABLE

In this section we will implement norms using anonymous games, and use this to establish the main result, namely that model checking $\mathcal{L}_{\text{NCHATL}}$ against Definition 3.4 takes polynomial time in the number of agents. We only show this for individual norms η, but due to the representation given in Equation 5.13, allowing us to represent any collective norm as an individual norm, the result for collective norms follows as an immediate corollary.

Now, for an individual normative system η and every $q \in Q$, we define the corresponding anonymous game $\mathsf{G}_\eta(q) = (\Sigma, \mathbb{A}_q, (\mu_i^q)_{i\in\Sigma})$ where for all $i \in \Sigma$ and every $s \in \mathbb{A}_q^\Sigma$ we take

$$\mu_i^q(s) = \begin{cases} 1 \text{ if } s(i) \in \mathbb{A}_q \setminus \eta_i(q, (\#(y, s_{-i}))_{y\in\mathbb{A}}) \\ 0 \text{ otherwise} \end{cases} \tag{5.19}$$

The following proposition follows easily, and we omit the proof.

PROPOSITION 4.1: *Given an* HCGS H *and a state* $q \in Q$, $s \in \mathbb{A}_q^\Sigma$ *is a Nash equilibrium for* $\mathsf{G}_\eta(q)$ *if, and only if,* $s \in \mathsf{L}_\eta(q)$, *where* $\mathsf{L}_\eta(q)$ *is as defined in Equation 5.10.*

In light of this fact, the following becomes a corollary of the fact that computing Nash Equilibria in anonymous games is tractable.

THEOREM 4.1: *Computing* $P_\eta(q)$ *takes polynomial time (actually, is in* TC_0*).*

Proof. It follows from Proposition 4.1 and Equation 5.11 that the set $P_\eta(q)$ is the set of profiles that can be induced by a Nash Equilibrium of $\mathsf{G}_\eta(q)$. Since this set can be computed in polynomial time using the characterization provided in Equation 5.8, the result follows. (For the claim in the parentheses, we refer to [22] who show that computing the set of profiles for Nash Equilibrium of an anonymous game is in the complexity class TC_0). □

We now demonstrate that partial compliance can be reasoned with in the same efficient manner as full compliance. That is, we must be able to compute the set $P_\eta^C(q)$ defined in Equation 5.16 in polynomial time for all $q \in Q$, $C \subseteq \Sigma$. Given the correspondence to anonymous games, we can do this if we can answer the following question in polynomial time: *given an anonymous game and a C-profile F, can F be induced by a partial action-tuple for C players, s, such that s can be extended to a Nash Equilibrium?*.

Notice that a C-profile need not have this property even if it can be extended to a profile that is induced by a Nash Equilibrium. If the game is homogeneous this holds trivially, but for anonymous games we cannot arbitrarily permute actions in the Nash Equilibrium to ensure that the desired C-profile is induced by explicit actions for members of C, and not just actions for members of some other coalition of the same size. Notice that this corresponds exactly to our observation about the special care that was needed in defining partial compliance to individual norms.

For all anonymous games $\Gamma = (\Sigma, \mathbb{A}, (\mu_i)_{i\in\Sigma})$ and all $C \subseteq \Sigma$ we say that $F \in P^C(\mathbb{A})$ is a partial Nash profile for Γ whenever

$$\exists s \in \mathsf{NE}(\Gamma) : (\#(y, s \restriction C))_{y\in\mathbb{A}} = F \tag{5.20}$$

Then the problem we must solve is the problem of deciding whether a given $F \in P^C(\mathbb{A})$ satisfies Equation 5.20. An answer to this suffices to

establish our main result since the notion of a partial Nash profile corresponds to partial compliance in the obvious way, c.f., Proposition 4.1. In particular, we have the following.

LEMMA 4.1: *For any* HCGS H*, any individual norm* η *and any* $q \in Q$*, we have* $F \in P^q_\eta(C)$ *if, and only if,* $F \in P^q(C)$ *is a partial Nash profile for* $\mathsf{G}_\eta(q)$*.*

Testing by brute force whether Equation 5.20 holds for some $F \in P^C(\mathbb{A})$ requires us to consider the potentially exponential set of all Nash Equilibria for the game Γ. However, we can simplify this by making another application of Hall's Lemma, analogous to that used to establish Equation 5.8. In particular, it is not hard to show that $F \in P^C(\mathbb{A})$ is a partial Nash profile for Γ if, and only if, there exists some $F' \in P^{\Sigma\setminus C}(\mathbb{A})$ such that the following two conditions hold:

$$\begin{aligned}
(1): \quad & \forall P \subseteq \mathbb{A}: \\
& |\{i \in C \mid \exists p \in P : p \in \mathsf{b}_i(s_{F+F'})\}| \geq \sum_{p\in P} F(p) \\
(2): \quad & \forall P \subseteq \mathbb{A}: \\
& |\{i \in \Sigma \setminus C \mid \exists p \in P : p \in \mathsf{b}_i(s_{F+F'})\}| \geq \sum_{p\in P} F'(p)
\end{aligned} \tag{5.21}$$

Using Equation 5.7 these conditions can be checked in polynomial time in the number of agents. Moreover, remember that running through different C-profiles, functions from $\mathbb{A}$ to $|C|$, is tractable by brute force when the number of actions remains fixed. Hence our main technical result follows.

THEOREM 4.2: *For any* HCGS H*, any individual norm* η *and any* $q \in Q$*, we can compute* $P^C_\eta(q)$ *in polynomial time in the number of agents.*

From this it follows easily that model checking is tractable in the agents. In particular, a simple adaptation of the model checking algorithm presented in [80], where we quantify over $P^C_\chi(q, D)$ in place of $P^C(q)$, witnesses to the truth of the following Theorem.

THEOREM 4.3: *Given an* HCGS H*, a norm* $\chi \in \{\eta, \kappa\}$*, a* $\mathcal{L}_{\text{NCHATL}}$*-formula* φ *a state* $q \in Q$ *and a coalition* $C \subseteq \Sigma$*, checking* $H, \chi, C, q \models \varphi$ *takes polynomial time in the number of agents.*

5 CONCLUSIONS AND FUTURE WORK

We studied normative systems for concurrent game structures, exploiting recent ideas from algorithmic game theory to ensure both tractability and greater expressiveness of the norms we consider. The tractability result was made possible by introducing homogeneity on the level of the system description, but as we showed, heterogeneous properties of interacting agents may be regained in a structured way using norms. Hence we provide a new perspective on where it is appropriate to encode heterogeneous properties of agents. For complexity reasons, introducing het-

erogeneity as a non-primitive notion, allowing it to arise in a limited way from mechanisms that aim to describe certain types of interactions between autonomous agents, might be more appropriate than embracing heterogeneity as an irreducible modeling assumption.

This perspective on heterogeneity also touches on points that may have conceptual significance above and beyond issues to do with complexity. In many cases homogeneity and/or anonymity arise naturally, for reasons to do with privacy, fairness, or lack of information. We point to [78] for a further discussion on the conceptual significance of the notion of homogeneity and anonymity in logics for multi-agent systems.

An interesting direction for future research is to investigate connections with cooperative game theory and social choice theory. In this paper we used normal form games to characterize and provide an incentive for a very simple *binary* notion of compliance for both individuals and coalitions. Intuitively, the notion of compliance we formalized was built on the idea that a coalition can be said to comply if, and only if, it acts in such a way that it is possible for all the remaining agents to act in accordance with the norm. A very natural next step is to consider *degrees* of compliance, where coalitions are rewarded on a gliding scale depending on how *easy* it becomes for the rest of the agents to fulfill the norm. This, in turn, leads us to cooperative game theory, where the norms themselves can give agents an *incentive* to form coalitions, to ensure higher rewards from better quality of compliance.

On the technical side, we hope to generalize $\mathcal{L}_{\text{NCHATL}}$ to allow quantification over coalitions, following the approach of [8]. For the structures considered there, the decision problems that arise after introducing such quantification are generally not tractable. For homogeneous structures, on the other hand, quantifying over coalitions should be possible, even in the context of broader norms, by relying on a compact representation. Essentially, a partitioning of coalitions into a limited number of equivalence classes would ensure efficient model-checking procedures also when quantification is involved.

We also hope to investigate more closely the link with non-homogeneous structures. In particular, it seems that we may in many cases be able to recognize different degrees of homogeneity within systems that are already formulated using heterogeneous means. Transforming such models to homogeneous models, using norms to model heterogeneous properties, is a challenge that will be considered.

In conclusion, we think broad norms on homogeneous structures provide a logical formalism for modeling multi-agent systems that have many attractive features, making it a suitable template for continued formal work on norms and strategic interaction.

ACKNOWLEDGMENTS Piotr Kaźmierczak's research was supported by the Research Council of Norway project number 194521 (FORMGRID).

6

PAPER C: COMPLIANCE GAMES

This paper was submitted to the 12th European Conference on Multi-Agent Systems (EUMAS 2014), taking place on December 18–19, 2014 in Prague, Czech Republic. At the time of writing this thesis, the paper was under review.

COMPLIANCE GAMES

Piotr Kaźmierczak
Dept. of Computing, Mathematics and Physics
Bergen University College, Norway

Abstract

In this paper we analyze *compliance games*, which are games induced by agent-labeled Kripke structures, goal formulas in the language of CTL and behavioral constraints. In compliance games, players are rewarded for achieving their goals while complying to social laws, and punished for non-compliance. Design of these games is an attempt at capturing notion of *sanctions* and incentivizing agents to be compliant. We analyze solution concepts and properties of compliance games, study the connection between underlying logical framework and their properties, and provide short analysis of key computational problems.

1 INTRODUCTION

Normative systems or *social laws*[1] are a framework for coordinating agents in multi-agent systems initially proposed by Shoham and Tennenholtz in [95, 96]. The idea has been extensively studied in the multi-agent systems literature since. While in Shoham and Tennenholtz's seminal papers the framework consisted of synchronous transition systems with first order logic language for goals, in further work other semantic structures and goal languages were used. In a series of papers, Ågotnes et al. [5, 8, 7, 2, 9] presented social laws implemented on agent-labeled Kripke structures with Computation Tree Logic (CTL) as a language for goals, while Van der Hoek et al. [102] used Alternating-time Transition Systems with Alternating-time Temporal Logic (ATL) [12], and a similar framework was used in [79, 42].

Each of these approaches uses the same idea, namely that we impose *restrictions* on agents' behavior,[2] and check which goals (expressed by our language of choice) are satisfied when agents comply with these restrictions. Social laws are usually studied as a higher level framework, abstracting away from institutions, sanctions and penalties, or run-time norm changes – in all the papers referenced above authors concentrate mostly

1 In the multi-agent systems literature normative systems and social laws stand for the same. However, in this paper we will always use the notion of a social law, since calling our restrictions "normative systems" can perhaps be confusing for readers with a background in deontic logic.

2 Thus social laws are sometimes also called "behavioral constraints."

on the *semantic* aspect of social laws, and not on law *specification* [21], as is often analyzed in deontic logic literature.

A number of interesting decision problems are usually studied in the social laws literature, such as *compliance sufficiency* (given a structure, a set of constraints, and a goal, which coalition's compliance to the constraints is sufficient in order for the goal to be achieved?), *k-robustness* (how many agents can deviate from complying to the normative system and still not break goal satisfiability?), *feasibility* (is it feasible for the agents to satisfy their goals while complying with the restrictions?), or social law *synthesis* (can we synthesize a set of restrictions such that when complied with they guarantee goal satisfaction?). Furthermore, Ågotnes et al. studied a number of meta-problems related to social laws, such as whether the laws are cost-optimal [2] or which coalitions have the most influence [7].

However, the key problem in social laws is how to assure that agents comply with a given social law. Our approach here is to introduce *penalties* or *sanctions*, i.e. to make compliance the rational choice for our agents. While in principle similar to the games presented by Ågotnes et al. in [5] where agents had preferences over goals and normal form games were induced based on the *utility* of laws, we extend the work presented in [61] and employ cooperative games to incentivize agents. The mechanism is simple: agents are rewarded for achieving goals while complying with laws, and punished (by means of null payoffs) for non-compliance. Formally, compliance games remain very simple and are induced by well-known agent-labeled Kripke structures, goal formulas expressed in the language of CTL and social laws understood as black-listed transitions of the Kripke structure. Our main contribution here is the representation theorem for compliance games and the analysis of stability (the core), which is a particularly problematic concept in this formal setting.

The paper is structured as follows. In Section 2 we provide the necessary formal background, Section 3 presents main definition of compliance games together with analysis of their properties, in Section 4 we discuss solution concepts, and in Section 5 we analyze complexity of decision problems. We conclude and discuss future work in Section 6.

2 TECHNICAL BACKGROUND

We begin by concisely presenting all the formal background for our work. This paper brings temporal logic, cooperative game theory and social laws together, thus we will present a rather concise introduction to all the necessary technicalities. First we describe the logical framework for evaluating objective formulas, then we explain the idea of social laws in detail and finally we introduce some concepts from cooperative game theory that we will need in later sections.

2.1 *Kripke structures and CTL*

We start by defining agent-labelled Kripke structures, in the same way as defined by Ågotnes et al. in [8]:

DEFINITION 2.1 (AGENT-LABELLED KRIPKE STRUCTURE): *An* agent-labeled Kripke structure *(henceforth referred to simply as* Kripke structure*) K is a tuple $\langle S, S^0, R, V, \Phi, \mathcal{A}, \alpha \rangle$ where:*

- *S is the non-empty, finite set of* states *and S^0 is the initial state,*
- *$R \subseteq S \times S$ is the serial[3] relation between elements of S that captures* transitions *between states,*
- *Φ is a non-empty, finite set of propositional symbols,*
- *$V : S \to 2^{\Phi}$ is a labeling function which assigns propositions to states in which they are satisfied,*
- *$\mathcal{A}$ is a non-empty finite set of agents, and*
- *$\alpha : R \to A$ is a function that labels edges with agents.[4]*

A *path* π over a relation R is an infinite sequence of states $s_0, s_1, s_2, \ldots$ such that $\forall u \in \mathbb{N} : (s_u, s_{u+1}) \in R$. $\pi[0]$ denotes the first element of the sequence, $\pi[1]$ the second, and so on. An *s-path* is a path π such that $\pi[0] = s$. $\Pi_R(s)$ is the set of s-paths over R, and we write $\Pi(s)$, if R is clear from the context.

Objectives are specified using the language of *Computation Tree Logic* (CTL), a popular branching-time temporal logic. We use an adequate fragment of the language defined by the following grammar:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{E}\bigcirc \varphi \mid \mathsf{E}(\varphi \mathcal{U} \varphi) \mid \mathsf{A}(\varphi \mathcal{U} \varphi)$$

where p is a propositional symbol. The standard derived propositional connectives are used, in addition to standard derived CTL connectives such as $\mathsf{A}\bigcirc \varphi$ for $\neg\mathsf{E}\bigcirc \neg\varphi$ (see [43] for details). We distinguish two fragments of the language defined above – a *universal* L^u (with a typical element u) and an *existential* L^e (with a typical element e) one:

$$u ::= \top \mid \bot \mid p \mid \neg p \mid u \vee u \mid u \wedge u \mid \mathsf{A}\bigcirc u \mid \mathsf{A}\Box u \mid \mathsf{A}(u \mathcal{U} u)$$
$$e ::= \top \mid \bot \mid p \mid \neg p \mid e \vee e \mid e \wedge e \mid \mathsf{E}\bigcirc e \mid \mathsf{E}\Box e \mid \mathsf{E}(e \mathcal{U} e)$$

Say that we are given two Kripke structures: $K_1 = \langle S, S^0, R_1, V, \Phi, A, \alpha \rangle$ and $K_2 = \langle S, S^0, R_2, V, \Phi, A, \alpha \rangle$. We say that K_1 is a subsystem of K_2 and K_2 is a supersystem of K_1 (denoted $K_1 \sqsubseteq K_2$) if and only if $R_1 \subseteq R_2$. This yields the following observation which we will later use to prove some properties of our games.

3 That is, $\forall s \exists t \ (s, t) \in R$.

4 While formally not necessary, throughout the paper we assume that an agent has to "own" at least one transition.

THEOREM 2.1 ([102]): *If $K_1 \sqsubseteq K_2$ and $s \in S$, then:*

$$\begin{array}{lll} \forall e \in L^e : K_1, s \models e & \Rightarrow & K_2, s \models e; \text{ and} \\ \forall u \in L^u : K_2, s \models u & \Rightarrow & K_1, s \models u. \end{array}$$

Satisfaction of a formula φ in a state s of a structure K, $K, s \models \varphi$, is defined as follows:

$$\begin{array}{l} K, s \models \top; \\ K, s \models p \text{ iff } p \in V(s); \\ K, s \models \neg\varphi \text{ iff not } K, s \models \varphi; \\ K, s \models \varphi \vee \psi \text{ iff } K, s \models \varphi \text{ or } K, s \models \psi; \\ K, s \models \mathsf{E}\bigcirc \varphi \text{ iff } \exists \pi \in \Pi(s) : K, \pi[1] \models \varphi; \\ K, s \models \mathsf{E}(\varphi \mathcal{U} \psi) \text{ iff } \exists \pi \in \Pi(s), \exists i \in \mathbb{N}, \text{s.t. } K, \pi[i] \models \psi \\ \qquad\qquad \text{and } \forall j, (0 \leq j < i) : K, \pi[j] \models \varphi; \\ K, s \models \mathsf{A}(\varphi \mathcal{U} \psi) \text{ iff } \forall \pi \in \Pi(s), \exists i \in \mathbb{N}, \text{s.t. } K, \pi[i] \models \psi \\ \qquad\qquad \text{and } \forall j, (0 \leq j < i) : K, \pi[j] \models \varphi. \end{array}$$

2.2 *Social laws*

A *social law* $\eta \subseteq R$ is a set of black-listed ("illegal") transitions, such that $R \setminus \eta$ remains serial.[5] The set of all social laws over R is denoted as $N(R)$. We say that $K \dagger \eta$ is a structure with a social law η *implemented* on it, i.e. for $K = \langle S, R, \Phi, V, A, \alpha \rangle$ and η, $K \dagger \eta = K'$ iff $K' = \langle S, R', \Phi, V, A, \alpha' \rangle$ with $R' = R \setminus \eta$ and:

$$\alpha'(s, s') = \begin{cases} \alpha(s, s') & \text{if } (s, s') \in R' \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Also, $\eta \restriction C = \{(s, s') : (s, s') \in \eta \ \&\ \alpha(s, s') \in C\}$ for any $C \subseteq A$ – that is to account for agents that do not necessarily comply with the social law (i.e. we can consider a situation in which only those edges that are "owned" by members of C are blacklisted).

EXAMPLE 2.1: *We introduce a running example that illustrates modeling a very simple Kripke structure.*

Figure 6.1 presents an example Kripke structure with:

- $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$, $S^0 = s_0$,
- $R = \{(s_0, s_0), (s_0, s_1), (s_1, s_2), (s_2, s_7), \ldots\}$,
- $\Phi = \{p, q\}$,

5 This is a so-called "reasonableness" constraint – we do not want social laws implementation of which results in systems with dead-end states.

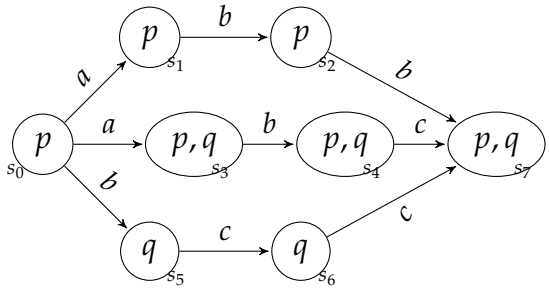

Figure 6.1: Simple Kripke structure example (reflexive loops omitted).

- $V(s_0) = \{p\}$, $V(s_1) = \{p\}, \ldots, V(s_7) = \{p, q\}$,
- $A = \{a, b, c\}$,
- $\alpha(s_0, s_1) = \alpha(s_0, s_3) = \{a\}$,
  $\alpha(s_1, s_2) = \alpha(s_3, s_4) = \alpha(s_0, s_5) = \alpha(s_2, s_7) = \{b\}$,
  $\alpha(s_4, s_7) = \alpha(s_5, s_6) = \alpha(s_6, s_7) = \{c\}$.

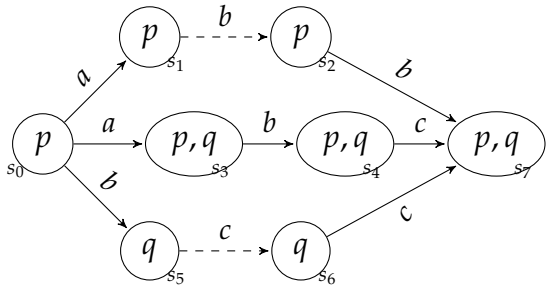

Figure 6.2: Kripke structure with a social law η (transitions in η are the dashed lines, again we omit reflexive loops for the sake of readability).

Figure 6.2 presents the same structure, but with a social law $\eta = \{(s_1, s_2), (s_5, s_6)\}$ *implemented on it.*

2.3 *Cooperative games*

We now introduce some concepts from cooperative game theory. Again, definitions provided here, albeit complete, are necessarily terse. For a more detailed explanation of the concepts introduced below, see [28].

DEFINITION 2.2: *A* cooperative game *(sometimes also called a* coalitional, *or* characteristic function *game) is a tuple* $G = \langle N, v \rangle$*, where* N *is a non-empty set of* players*, and* $v : 2^N \to \mathbb{R}$ *is a* characteristic function *of the game which assigns a* value *to each coalition* $C \subseteq N$ *of players.*[6]

6 Cooperative games considered here are *transferable utility* (TU) games, which means that the characteristic function v assigns value to coalitions, but does not say anything about the division of payoffs to individual players.

We say a cooperative game $G = \langle N, v \rangle$ is *monotone* (or *increasing*) if $v(C) \leq v(D)$ whenever $C \subseteq D$ for $C, D \subseteq N$. A cooperative game is *simple* when each of the coalitions of players either wins or loses the game, in other words, when the characteristic function's signature is $v : 2^N \to \{0, 1\}$.

Finally, we say that player i is a *veto player* in game G if $v(C) = 0$ for any $C \subseteq N \setminus \{i\}$, and that i is a *dummy player* if $v(C) = v(C \cup \{i\})$ for any $C \subseteq N$.

3 COMPLIANCE GAMES

We now consider cooperative games in which agents are rewarded for satisfying formulas and punished for violating laws.

We evaluate agents' actions based on how many of their respective *goal* formulas they are able to satisfy. Thus we say that, given a Kripke structure K, there is a set of goals:

$$\gamma_i = \{\varphi_1, \ldots, \varphi_m\}$$

associated with each agent $i \in A$ of K, where φ_j is a *goal formula* expressed in the language of CTL. We say that a Kripke structure K, a set of goals γ_i for each agent $i \in A$ of K and a social law η over R of K constitute a *social system* $\mathcal{S} = \langle K, \gamma_1, \ldots, \gamma_n, \eta \rangle$.

Below we introduce the definition of the game. The idea behind it is that the value of a coalition C is the amount of goals it can achieve under restrictions minus the amount of goals achievable in a Kripke structure (without restrictions). This number can be negative, and the rationale behind such design of games is that behavior of agents would indicate to the system designer whether the laws he designed are optimal or not (i.e., if an agent can satisfy more of his goals while not complying than when complying then perhaps either the goals or the laws need to be adjusted).

DEFINITION 3.1 (COMPLIANCE GAME): *A social system $\mathcal{S}$ induces a cooperative game $G_{\mathcal{S}} = \langle \mathcal{A}, v_{\mathcal{S}} \rangle$, where $\mathcal{A}$ is a set of agents in K of $\mathcal{S}$, and*

$$v_{\mathcal{S}}(C) = \begin{cases} 1 & \text{if } \left(\sum_{\varphi \in \gamma_C} \left| \{\varphi : K \dagger (\eta \restriction C) \models \varphi\} \right| - \sum_{\varphi \in \gamma_C} \left| \{\varphi : K \models \varphi\} \right| \right) > 0 \\ 0 & \text{otherwise,} \end{cases}$$

where $C \subseteq \mathcal{A}$, and $\gamma_C = \bigcup_{i \in C} \gamma_i$.

We now analyze some properties of compliance games. First we observe that the characteristic function of said games is not always monotone.

LEMMA 3.1: *Compliance games are not always monotone.*

Proof. The lemma above can be proved with a simple counter example shown in Figure 6.3. As seen in the example, adding agents to a winning coalition can "break" the satisfiability of their goals. □

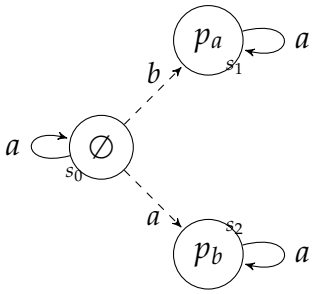

Figure 6.3: A Kripke structure which induces a non-monotone compliance game, illustrating Lemma 3.1. Here, $\gamma_a = \{\mathsf{E}\bigcirc p_a\}$, $\gamma_b = \{\mathsf{E}\bigcirc p_b\}$, $K \models \gamma_a$, $K \models \gamma_b$, and $K \dagger (\eta \restriction \{a\}) \models \gamma_a$, $K \dagger (\eta \restriction \{b\}) \models \gamma_b$, but $K \dagger (\eta \restriction \{a, b\}) \not\models \gamma_a \vee \gamma_b$.

3.1 *Representation theorem*

The fact that compliance games are not always monotone is a negative result from a point of view of algorithmic game theory, because we cannot take computational advantage of the monotonicity property of the characteristic function. In fact we present a Theorem below which states something even stronger:

THEOREM 3.1: *Given an arbitrary function $v : 2^{\mathcal{A}} \to \{0, 1\}$, there is always a social system $\mathcal{S} = \langle K, \gamma_1, \ldots, \gamma_n, \eta \rangle$ which induces a cooperative game $G_{\mathcal{S}} = \langle A, v_{\mathcal{S}} \rangle$, where A is a set of agents in K of $\mathcal{S}$ and $v = v_{\mathcal{S}}$.*

Proof. We prove the theorem by providing a recipe for creating a social system $\mathcal{S} = \langle K, \gamma_1, \ldots, \gamma_n, \eta \rangle$ in which K is constructed of elements which "isolate" winning conditions for each winning coalition.

We construct $\mathcal{S}$ in the following way. Each agent is given the same goal: $\mathsf{E}\Diamond\mathsf{A}\Box p$, thus γ_i is a singleton set which contains one formula. We then construct the Kripke structure K starting from the initial state which is not labeled by any proposition and has a transition labeled by an arbitrary agent and not blacklisted by η (a "bridge") leading to another state s_C (again, not labeled by any proposition). The s_C state is the beginning of a construction which assures that coalition C wins. We then construct a sequence of states not labeled by any propositions with transitions labeled by all agents from $\mathcal{A} \setminus C$, one per agent, all of which are included in the social law η (this assures that the superset of C does not win along this path). Once we are done, we add a state labeled by all the goals of members of C and a reflexive loop labeled by an arbitrary agent. This state becomes the satisfied goal once members of C comply to η. Next, in order to assure subsets of C lose the game, we add a state and a transition per member of C leading to a state in which the negation of all the goals is satisfied, labeling transitions with respective agents and adding them to η – this way only if all members of C comply the transitions will be blacklisted and the goal satisfied. This whole construction from s_0 assures winning

conditions for coalition C, and we can construct a separate such construction for each winning C'. Any C' that does not have a construction of this kind is a losing coalition, thus we can model *any* set of outcomes of the game. The construction is presented in Figure 6.4. □

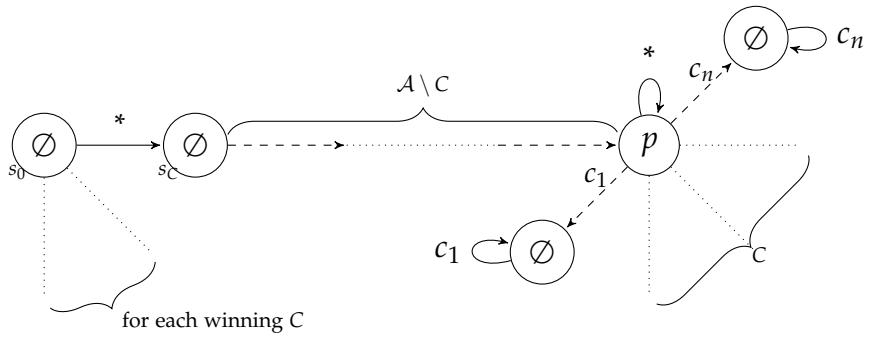

Figure 6.4: Construction for the proof of Theorem 3.1. Dashed lines represent transitions in η, the $*$ symbol stands for an arbitrary agent.

EXAMPLE 3.1: *To illustrate the construction presented in Theorem 3.1 we provide a simple example of a social system $\mathcal{S}$ with three agents a, b and c, sharing one goal formula $\varphi = \mathsf{E}\Diamond\mathsf{A}\Box p$ and inducing a compliance game $G_{\mathcal{S}} = \langle \mathcal{A}, v \rangle$ where the outcome of $v(C)$ is as follows:*

$$\begin{aligned} v(\{a\}) &= 1 \\ v(\{b\}) = v(\{c\}) = v(\{a,b\}) = v(\{b,c\}) = v(\{a,c\}) &= 0 \\ v(\mathcal{A}) &= 1 \end{aligned}$$

The social system $\mathcal{S}$ is presented in Figure 6.5. The figure presents two separate constructions from Theorem 3.1, one for each winning coalition ($\{a\}$ and the grand coalition). Notice the right branch of the tree does not contain any blacklisted transitions from the "bridge", since it implements winning conditions for the grand coalition.

The representation result presented in this section is more general, since it can easily be adapted to similar simple games. In [7], authors present social systems $S = \langle K, \varphi, \eta \rangle$ and induce simple cooperative games of the form

$$v_S(C) = \begin{cases} 1 & \text{if } K \dagger (\eta \restriction C) \models \varphi \\ 0 & \text{otherwise,} \end{cases}$$

in order to study power indices for agents. Since the game defined above is very similar to our compliance game, we immediately obtain the following result:

COROLLARY 3.1: *Given an arbitrary function $v : 2^{\mathcal{A}} \to \{0,1\}$, there is always a social system $S = \langle K, \varphi, \eta \rangle$ which induces a cooperative game $G_S = \langle \mathcal{A}, v_S \rangle$ as defined in [7], where $\mathcal{A}$ is a set of agents in K of S.*

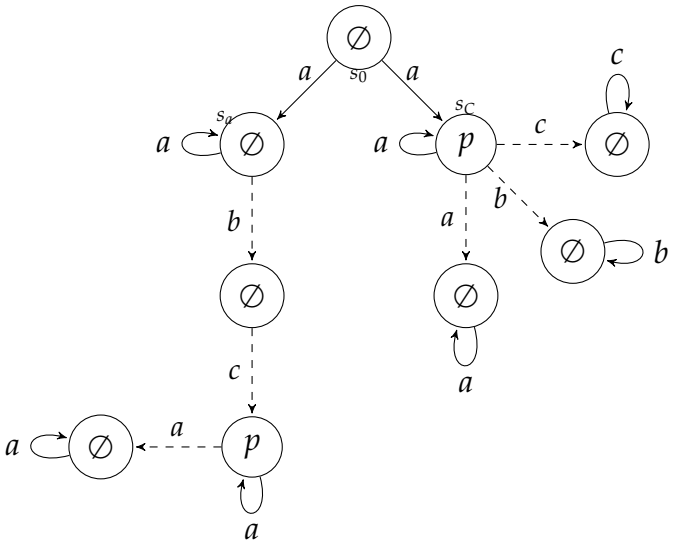

Figure 6.5: Example social system that illustrates the construction from Theorem 3.1.

Since monotonicity is a desirable property for most coalitional games, we are interested in identifying subclasses of our games that are actually monotone. Ågotnes et al. [8] identify social systems with universal goals as a subclass of monotone versions of their games. We can do the same for our compliance games.

PROPOSITION 3.1: *Compliance games are monotone for $\varphi \in L^u$.*

Proof. Let $\mathcal{S} = \langle K, \eta, \gamma_1, \ldots, \gamma_n \rangle$ be a social system, and let $C \subseteq C' \subseteq \mathcal{A}$ be coalitions in K. Then from Theorem 2.1 we know that if $K \dagger (\eta \restriction C) \models u$ then $K \dagger (\eta \restriction C') \models u$. Thus if an agent's universal goal becomes satisfied by his compliance (removal of edges), another agent's compliance cannot break satisfiability of that goal. For the same reason if there is a universal goal satisfied in a Kripke structure, imposing a social law will not change its satisfiability. □

4 SOLUTION CONCEPTS

Cooperative game solution concepts can in general be divided into two categories: ones that are concerned with *stability*, and ones that are concerned with *fairness*. The first category of solution concepts attempts to characterize conditions for games in which there is a stable coalition structure, i.e. no players have an incentive to abandon their respective coalitions and form new ones. The second category attempts to calculate the value of players' contributions to their respective coalitions and allocate payoffs in a "fair" way – players that contribute more are rewarded.

In this section we discuss how both these kinds of solution concepts can be applied to compliance games, starting with stability.

4.1 *Stability – the core*

We concentrate on the most popular stability related solution concept, which is *the core*. In order to define it precisely, we need a few extra definitions.

We say that an *imputation* is a vector $(p_1, \ldots, p_n)$ with $p_i \in \mathbb{Q}$ which is such a division of gains of a grand coalition N that $\sum_{i=1}^{n} p_i = v(N)$. We say that p_i is a *payoff* for player i and denote the payoff for coalition C as $p(C) = \sum_{i \in C} p_i$.

An imputation satisfies *individual rationality* when for all players $i \in C$, we have $p_i \geq v(\{i\})$. A coalition B *blocks* a payoff vector $(p_1, \ldots, p_n)$ when $p(B) < v(B)$ – members of B can abandon the original coalition, with each member getting p_i and there is still some amount of $v(B)$ to be shared amongst players. The coalition is thus unstable when a blocking payoff vector is chosen.

DEFINITION 4.1: *The* core *of a cooperative game is a set of such imputations which are not blocked by any coalition. Formally, for an imputation $(p_1, \ldots, p_n)$, and any coalition C, $p(C) \geq v(C)$.*

Intuitively, the core characterizes such a set of outcomes where no player has an incentive to abandon the coalition structure. Many games have an empty core, however simple games usually have a non-empty core, due to the following well-known result:

LEMMA 4.1 ([75]): *A core of a simple cooperative game is non-empty iff there is at least one veto player in the game. And if there are veto players, any imputation that distributes payoffs only to them is in the core of the game.*

Thus checking non-emptiness of the core for a simple game boils down to finding whether there are veto players.

Before we attempt to identify outcomes in the core of compliance games, we need to discuss in detail what "stability" of said games actually means.

Recall that in Definition 3.1 the value for a coalition is defined as the difference between how many goals a coalition can achieve while complying to a social law and the amount of goals achievable when not complying. There is an important detail in the semantics of $K \dagger (\eta \restriction C)$ expression, though, that makes an interpretation of stability-capturing solution concepts somewhat problematic.

As mentioned in the paragraphs above, the core characterizes a set of stable outcomes, where stable means players have no incentives to "deviate" and pursue goals on their own. What does "deviating" mean in the context of compliance games? It means complying, but with other players. However, the expression $K \dagger (\eta \restriction C)$ means that players in coalition C comply with η, but at the same time *no other players comply*. This means that if a proper subset of coalition C intends to abandon its coalition and form a new coalition C', the rationale behind its agents' behavior is to form a coalition in which they themselves comply, and at the same time

they somehow force all other players to *not* comply. This aspect of compliance games is highly unusual and makes the interpretation of the core non-standard.

The way we motivate and explain such design of compliance games is that we are interested in said games from a system designer's point of view. Reasoning this way, we may interpret a particular coalition structure of a compliance game as a set of *hypothetical scenarios*: each winning coalition is such a scenario, and in this scenario the coalition in question assumes no one but its members comply. We could say winning coalitions are "selfishly winning", and then the core of a compliance game would correspond to a situation in which making less *or more* agents comply would break the coalition structure. We feel such an interpretation of stability and the core is an acceptable one, but the reader should keep in mind it is not a standard interpretation.

PROPOSITION 4.1: *Compliance games can have empty cores.*

Proof. From Lemma 4.1 we know that the only situation when a compliance game has an empty core is when there are no veto players present in the game. From Theorem 3.1 we know that we can create a social system that induces a compliance game with a characteristic function that has an arbitrary output, so e.g. for a social system with two players a and b we can construct $v(\{a\}) = v(\{b\}) = 1$, but $v(\{a, b\}) = 0$ and $v(\mathcal{A}) = 0$. □

In light of this negative result we would be interested in a relationship between the shape of elements of $\mathcal{S}$ and the non-emptiness of the core of the compliance game it induces, or more narrowly, we are interested in the relationship between the structure of $\mathcal{S}$ and the existence of veto players.

The first likely suspects to yield games with non-empty cores are social systems with only universal goal formulas, since these games will be monotone, as shown in Proposition 3.1. In monotone games it is easy to see that player i is a veto player iff $v(\mathcal{A} \setminus \{i\}) = 0$, thus any game with at least one winning and one losing coalition will have at least one veto player.

PROPOSITION 4.2: *Let $\mathcal{S}$ be a social system containing only universal goals. If $\mathcal{S}$ induces a compliance game where there is at least one winning and one losing coalition, then this game's core is non-empty.*

Proof. Follows directly from the statement of the problem and Proposition 3.1. □

4.2 *Fairness – Shapley value*

We now move on to discussing *Shapley value* [92] – a solution concept that captures the notion of fairness. Intuitively, Shapley value measures agents'

marginal contribution depending on *when* they join a coalition and pays proportionally to that contribution.

Let $G = \langle N, v \rangle$ be a cooperative game. We say Π_N is the set of all *permutations* of N. Given a permutation $\pi \in \Pi_N$ we denote $Pred_\pi(i)$ as the set of *predecessors* of i in π. The *marginal contribution* of agent i with respect to a permutation π in game G is then denoted as $\Delta^G_\pi(i)$ and defined as:

$$\Delta^G_\pi(i) = v(Pred_\pi(i) \cup \{i\}) - v(Pred_\pi(i)).$$

Intuitively, $\Delta^G_\pi(i)$ measures how much the value of a given coalition increases when i joins it. Thus the Shapley value of player i can be defined as an average marginal contribution over all permutations of the set of players. Formally:

DEFINITION 4.2 (SHAPLEY VALUE): *Given a cooperative game $G = \langle N, v \rangle$ where $|N| = n$,* Shapley value *of player $i \in N$ is denoted by $\mathsf{Sh}_i(G)$ and defined as:*

$$\mathsf{Sh}_i(G) = \frac{1}{n!} \sum_{\pi \in \Pi_N} \Delta^G_\pi(i)$$

Shapley value as defined in the classical version above applies to non-simple games, i.e. where the characteristic function can output any positive real number. Since our compliance games are simple, we could adopt a version of Shapley value for simple games, called the *Shapley-Shubik value* [93]. However, since our games resemble the games used by Ågotnes et al. to a great degree, we decide to adapt the games rather than adapt the solution concept. Thus we introduce a non-simple version of compliance games:

DEFINITION 4.3 (NON-SIMPLE COMPLIANCE GAME): *A social system $\mathcal{S}$ induces a cooperative game $G_\mathcal{S} = \langle \mathcal{A}, v_\mathcal{S} \rangle$, where $\mathcal{A}$ is a set of agents in K of $\mathcal{S}$, and:*

$$v_\mathcal{S}(C) = max\Big(0, \Big(\sum_{\varphi \in \gamma_C} \big|\{\varphi : K \dagger (\eta \restriction C) \models \varphi\}\big| - \sum_{\varphi \in \gamma_C} \big|\{\varphi : K \models \varphi\}\big| \Big)\Big)$$

where $C \subseteq \mathcal{A}$ and $\gamma_C = \bigcup_{i \in C} \gamma_i$.

In other words, we are now interested simply in a number of formulas coalitions can satisfy while complying and set their value to this amount. The following example illustrates calculating said values.

EXAMPLE 4.1: *Consider a game G with three players: $\mathcal{A} = \{a, b, c\}$ and two coalitions which are able to achieve goals while complying to the laws but not when not complying. Agent a can achieve two of his goals on his own this way, and the grand coalition can achieve three goals of its members. Hence, $v(\{a\}) = 2$, $v(\{a, b\}) = v(\{b, c\}) = v(\{a, c\}) = v(\{b\}) = v(\{c\}) = 0$, and $v(\mathcal{A}) = 3$. Let us calculate the Shapley value for player a in G.*

First we list all permutations of the set of agents:

$$\Pi_G = \{(a, b, c), (a, c, b), (b, a, c), (b, c, a), (c, a, b), (c, b, a)\}$$

and enumerate them $\pi_1, \ldots, \pi_6$. Then the marginal contribution of player a is as follows:

$$\Delta^G_{\pi_1}(a) = \nu(\{a\}) - \nu(\emptyset) = 2$$
$$\Delta^G_{\pi_2}(a) = \nu(\{a\}) - \nu(\emptyset) = 2$$
$$\Delta^G_{\pi_3}(a) = \nu(\{b,a\}) - \nu(\{b\}) = 0$$
$$\Delta^G_{\pi_4}(a) = \nu(\{b,c,a\}) - \nu(\{b,c\}) = 3$$
$$\Delta^G_{\pi_5}(a) = \nu(\{c,a\}) - \nu(\{c\}) = 0$$
$$\Delta^G_{\pi_6}(a) = \nu(\{c,b,a\}) - \nu(\{c,b\}) = 3$$

And thus the Shapley value of a, $\mathsf{Sh}_a(G) = 2+2+3+3/6 = \frac{5}{3}$.

When dealing with Shapley values we are interested in finding dummy players. Recall that we call player i in game $G = \langle N, \nu \rangle$ a dummy player when $\nu(C) = \nu(C \cup \{i\})$ for any $C \subseteq N$. A known result about dummy players is that their Shapley value is 0.

PROPOSITION 4.3 ([28]): *Let $G = \langle N, \nu \rangle$ be a cooperative game. If a player $i \in N$ is a dummy player, then* $\mathsf{Sh}_i(G) = 0$.

From a perspective of a system designer using the social laws paradigm to model his systems it is interesting to see how quickly he can identify dummy players – since it is known that the Shapley values for any cooperative game sum up to the value for the grand coalition,[7] knowing which players are dummy is useful.

5 COMPLEXITY

We now consider the computational complexity of several important decision problems. As usual in cooperative games, we are primarily interested in how fast are we able to check whether there are any veto (for computing the core of simple games) or dummy (for finding agents whose Shapley value is 0) players in our games.

The computational problems are defined formally below:

VETO: Given a compliance game $G_{\mathcal{S}}$ induced by a social system $\mathcal{S}$, and i a player in $G_{\mathcal{S}}$, is i a veto player?

DUMMY: Given a non-simple compliance game $G_{\mathcal{S}}$ induced by a social system $\mathcal{S}$, and i a player in $G_{\mathcal{S}}$, is i a dummy player?

Before we start proving complexity results about the problems defined above, we note that calculating the value of the characteristic function of both simple and non-simple compliance games can be done in time polynomial in the size of the goals. This is due to the fact that calculating value of that function is simply performing model checking of CTL formulas, and this problem is known to be solvable in polynomial time [32].

7 That is, given that the grand coalition actually forms, which is not necessarily true for compliance games.

PROPOSITION 5.1: VETO *is in co-*NP *in the general case, and polynomial-time solvable for universal goals.*

Proof. Showing that VETO is in co-NP can be easily done by analyzing the complement problem. Say we want to check that i is not a veto player. i is not a veto player if there is a coalition C not containing i for which $v(C) = 1$. Hence showing co-NP-membership is equivalent to being able in polynomial time to verify that some C does not contain i and that $v(C) = 1$. Thus, VETO is in co-NP. For universal goals compliance games are monotone, and then finding veto players means checking that $v(\mathcal{A} \setminus \{i\}) = 0$, which can be done in polynomial time. □

An immediate corollary of the above result is that checking non-emptiness of the core of simple compliance games is co-NP-complete, as we know from Lemma 4.1. It is, however, solvable in polynomial time for games with only universal goals.

PROPOSITION 5.2: DUMMY *is in co-*NP.

Proof. Membership is easy to see. If an agent is not a dummy player, then we can show this by finding a coalition for which $v(C) = k$, but $v(C \cup \{i\}) \neq k$. Thus, DUMMY is in co-NP. □

6 CONCLUSIONS AND FUTURE WORK

In this paper we presented a new, game theoretic approach to incentivizing agents' compliance to social laws. We analyzed properties of compliance games, studied solution concepts and relation between their properties and the underlying logical framework that induces them.

For future work, we plan to first investigate some tractable instances for our games. Complexity results for compliance games are negative (except for the case of universal-only goals), hence we would be interested in finding special cases of structures and perhaps some conditions for social laws which could induce monotone or submodular games. Further future work could involve investigating how we could use mechanism design for incentivizing agents to comply.

And finally, we plan to try designing compliance games based on other classes of cooperative games, such as Boolean games [41] (where every agent controls the value of a propositional formula), skill games [14] (in which each agent has a certain set of skills required to completing a task), or Qualitative Coalitional Games [110] (non-transferable utility games in which agents make choices and different sets of goals become achievable depending on these choices). We think that the general idea of using cooperative game theory for modeling compliance to social laws is a naturally attractive and to a great degree unexplored idea, and we wish to pursue it further in the future.

7

PAPER D: NORMC: A NORM COMPLIANCE TEMPORAL LOGIC MODEL CHECKER

This paper was presented during the Sixth Starting Artificial Intelligence Research Symposium (STAIRS 2012) in August 2012 in Montpellier, France. It has been published in the symposium proceedings by iOS Press.

NORMC: A NORM COMPLIANCE TEMPORAL LOGIC MODEL CHECKER

Piotr Kaźmierczak
Dept. of Computing, Mathematics and Physics
Bergen University College, Norway

Truls Pedersen
Dept. of Information Science and Media Studies
University of Bergen, Norway

Thomas Ågotnes
Dept. of Information Science and Media Studies
University of Bergen, Norway

Abstract

In this paper we describe NorMC, a model checker for Norm Compliance CTL, implemented in the Haskell programming language. NorMC is intended as a tool for students, researchers, and practitioners to learn about and understand normative systems, and as an exploratory tool for researchers in multi-agent systems. The objectives of the paper are twofold. First, to give a system description of NorMC. Second, to argue and demonstrate that the Haskell programming language is a natural and useful alternative for model checking; in particular that the full power of Haskell makes it easy to describe arbitrary state-transition models in a natural way.

1 INTRODUCTION

Normative systems, or *social laws*, have emerged as a promising and powerful framework for coordinating multi-agent systems [95, 96, 45, 102, 4, 5, 36, 7, 8]. The starting point is a state-transition model of a multi-agent system, typically a *legacy system*, and the goal is to constrain the behaviour of the agents in the system in such a way that the global behaviour of the system exhibits some desirable properties. Such a restriction on agents' behaviour is called a normative system, or a social law. The desirable global properties are typically represented using a (modal) logical formula; the *objective* (typically not satisfied in the initial system).

A key issue in normative systems is the question of *compliance*. Even if a normative system is *effective*, i.e., will ensure that the objective holds, under the assumption that all agents comply with it – how do we know that

they will comply? And what happens if they do not? There are several possible reasons for non-compliance [8]: an autonomous and rational agent might choose not to comply because it is not in her best interest; a rational agent might not comply by accident (a failure); an irrational agent might choose not to comply without any particular reason. *Norm compliance* CTL (NCCTL) [8] was developed to reason about normative systems and in particular about (non-)compliance. It can be used, e.g., to model check compliance properties such as "which agents *have to comply* for the objective to hold". NCCTL extends the branching-time temporal logic *Computation-Tree Logic* (CTL) [43] with a unary modality $[P]$, where P is a *coalition predicate*, i.e., a possible property of groups of agents (*coalitions*). The meaning of the expression $\langle P \rangle \varphi$ is that if any coalition that satisfies P complies with the normative system, then φ will hold. Examples of NCCTL expressions include the following, which are evaluated in the context of a model and a normative system:

- $[supseteq(C)]\varphi$: if any superset of C comply, φ will hold (C is *sufficient*)
- $[\neg geq(k)]\neg\varphi$: at least k agents have to comply for φ to hold (the normative system is *k-necessary*)
- $[geq(n-k)]\varphi \wedge \langle ceq(n-k-1)\rangle\neg\varphi$, where n is the total number of agents: k is the largest number of non-compliant agents the normative system can tolerate whilst still being effective for φ (the *resilience* of the normative system is k)

Formally, $[P]$ has a non-standard *update semantics*, which makes it difficult to use standard branching-time model checkers directly to verify normative system properties specified in NCCTL.

In this paper we describe NORMC, a prototype model checker for NCCTL implemented in Haskell [60]. The intended use of NORMC is as a tool to learn about and understand normative systems, and as an exploratory tool for researchers. Even small "toy" examples can be difficult to understand properly without a computational tool, because the number of model modifications (updates) resulting from different groups of agents complying is typically exponential in the size of the model. As a prototype tool intended for academic rather than industrial use, the focus in the implementation of NORMC has not been on efficiency, but rather on clarity, extensibility and ease of use. In particular, standard symbolic model checking optimisation techniques have not been implemented. However, the implementation is still efficient enough for interesting and non-trivial examples.

The objectives of the paper are twofold. First, the paper is a system description of NORMC that demonstrates how it can be used to model check normative systems. Second, we want to argue for and demonstrate the usefulness of Haskell programming language for model checking modal logics, as already suggested by model checkers such as the epistemic logic

model checker DEMO [103]. Haskell's native support for discrete structures and lazy evaluation mechanism makes it well suited for programming model checking algorithms. More importantly, that a model checker is implemented in Haskell means that it is possible to have the full power of the Haskell language available for the *user* to *describe models* – in contrast with the restricted model description languages available in popular temporal logic model checkers such as SPIN [52] or NUSMV [31].

The paper is organised as follows. We first review the formal normative systems framework and NCCTL. In Section 3 we describe how to specify a model and execute the model checker via a simple example. In Section 4 we illustrate further with a more complicated example, and show how NORMC was used to find an error in a case study in the literature. The implementation of NORMC is discussed in Section 5, and we conclude in Section 6. NORMC, with source code, can be downloaded from

http://pkazmierczak.github.com/NorMC/.

2 BACKGROUND

We give an, necessarily terse due to lack of space, overview of the background; see [8] for more details. Assume a set Φ of propositional variables. As the semantic model we use an *agent-labelled Kripke structure*; a tuple $K = \langle S, s^0, R, A, \alpha, V\rangle$ where S is a finite, non-empty set of *states*; $s^0 \in S$ is the *initial state*; $R \subseteq S \times S$ is a serial binary relation (i.e., $\forall s \exists t\ (s,t) \in R$) on S, which we refer to as the *transition relation*; A is a set of *agents*; $\alpha : R \to A$ labels each transition in R with an agent; and $V : S \to \wp(\Phi)$ labels each state with a set of propositional variables.

A *path* π over a relation R is an infinite sequence of states $s_0, s_1, s_2, \ldots$ such that $\forall u \in \mathbb{N} : (s_u, s_{u+1}) \in R$. $\pi[0]$ denotes the first element of the sequence, $\pi[1]$ the second, and so on. An *s-path* is a path π such that $\pi[0] = s$. $\Pi_R(s)$ is the set of s-paths over R, and we write $\Pi(s)$, if R is clear from the context.

To specify objectives we use CTL. We use an adequate fragment of the language defined by the following grammar:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{E} \bigcirc \varphi \mid \mathsf{E}(\varphi \mathcal{U} \varphi) \mid \mathsf{A}(\varphi \mathcal{U} \varphi)$$

where $p \in \Phi$. The standard derived propositional connectives are used, in addition to standard derived CTL connectives such as $A \bigcirc \varphi$ for $\neg E \bigcirc \neg\varphi$. Satisfaction is defined as follows (propositional clauses as usual):

$$\begin{aligned}
&K, s \models \mathsf{E} \bigcirc \varphi \text{ iff } \exists \pi \in \Pi(s) : K, \pi[1] \models \varphi;\\
&K, s \models \mathsf{E}(\varphi \mathcal{U} \psi) \text{ iff } \exists \pi \in \Pi(s), \exists u \in \mathbb{N}, \text{s.t. } K, \pi[u] \models \psi\\
&\qquad\qquad\qquad\qquad \text{and } \forall v, (0 \leq v < u) : K, \pi[v] \models \varphi;\\
&K, s \models \mathsf{A}(\varphi \mathcal{U} \psi) \text{ iff } \forall \pi \in \Pi(s), \exists u \in \mathbb{N}, \text{s.t. } K, \pi[u] \models \psi\\
&\qquad\qquad\qquad\qquad \text{and } \forall v, (0 \leq v < u) : K, \pi[v] \models \varphi;
\end{aligned}$$

A *normative system* η over K is a set of *constraints* on the behaviour of the agents. Formally, $\eta \subseteq R$, such that $R \setminus \eta$ is a total relation, represents the *forbidden* transitions. We say that normative systems are *implemented* on Kripke structures, which means that after the implementation all the transitions that are forbidden according to a normative system are removed from the structure. Formally, if η is a normative system over K, then $K \dagger \eta$ stands for the Kripke structure obtained from K by removing the transitions forbidden by η, i.e., if $K = \langle S, s^0, R, A, \alpha, V\rangle$ and $\eta \in N(R)$ (where $N(R)$ is all normative systems over K), then $K \dagger \eta = \langle S, s^0, R', A, \alpha', V\rangle$ where $R' = R \setminus \eta$, and α' is the restriction of α to R': if $(s, s') \in R'$ then $\alpha'(s, s') = \alpha(s, s')$. A set $C \subseteq A$ is a *coalition*. Let η be a normative system over K, then $\eta \restriction C$ is the normative system restricted to the actions of agents in C: $\eta \restriction C = \{(s, s') : (s, s') \in \eta \ \&\ \alpha(s, s') \in C\}$.

The language of NCCTL extends the language of CTL with a unary modality $\langle P\rangle$ where P is a *coalition predicate*. The language of coalition predicates is defined by the following grammar:

$$P ::= subseteq(C) \mid supseteq(C) \mid geq(n) \mid \neg P \mid P \vee P$$

where $C \subseteq A$ and $n \in \mathbb{N}$. Satisfaction of a predicate P by a coalition C_0, $C_0 \models_{cp} P$, is defined straightforwardly: $C_0 \models_{cp} subseteq(C)$ iff $C_0 \subseteq C$; $C_0 \models_{cp} geq(n)$ iff $|C_0| \geq n$; $C_0 \models_{cp} supseteq(C)$ iff $C_0 \supseteq C$; $C_0 \models_{cp} \neg P$ iff not $C_0 \models_{cp} P$; and $C_0 \models_{cp} P_1 \vee P_2$ iff $C_0 \models_{cp} P_1$ or $C_0 \models_{cp} P_2$.

Formally, the language of the NCCTL is generated as follows:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{E}\bigcirc\varphi \mid \mathsf{E}(\varphi \mathcal{U} \varphi) \mid \mathsf{A}(\varphi \mathcal{U} \varphi) \mid \langle P\rangle\varphi$$

We also write $[P]\varphi$ to denote the dual coalition predicate: $\neg\langle P\rangle\neg\varphi$.

Formulas of NCCTL are interpreted in a triple (K, η, s) where K is a Kripke structure, η a normative system over K and s is a state of K. The clauses for coalition predicates are as follows:

$$K, \eta, s \models \langle P\rangle\varphi \text{ iff } \exists C \subseteq A \ (C \models_{cp} P \text{ and } K \dagger (\eta \restriction C), \eta, s \models \varphi)$$

All other formulas are defined as for CTL, but carrying the normative system in the context: e.g. $K, \eta, s \models \mathsf{E}\bigcirc\varphi$ iff $\exists \pi \in \Pi(s) : K, \eta, \pi[1] \models \varphi$.

3 INTRODUCING NORMC BY EXAMPLE

In this section we demonstrate how NorMC is used, via a simple example. The core of the model checker is a function called `check`. By importing the NorMC code into a Haskell interpreter, this function can be used as follows:

```
*Example> check myModel myNS myFormula
[s1, s3, s7]
```

Here, the arguments `myModel`, `myNS` and `myFormula`, represent the model, the normative system, and the formula respectively, and the function returns

the set of (in this case three) states in which the formula holds in the given model and normative system. We now describe the types of these three arguments expected by `check`.

Kripke models are represented by the `Kripke` data structure. To represent sets, such as the set of states in the `Kripke` structure, sorted lists with no duplicate occurrences are used. In NORMC `Kripke` is defined as follows:

```
data (Ord s, Eq p) ⇒ Kripke p s = Kripke {
  agents :: [Int],       states :: [s],
  tr :: FODBR s s,       owner :: (s, s) → Int,
  valuation :: p → [s]  }
```

As an example, we describe a simple model of a pavement, divided into 10 *positions*, with two agents on opposite sides of it. The agents can move (or choose to stand still) alternatingly. The model, in its initial state, can be visualised as follows:

The model is defined by the following declaration.

```
exampleModel :: Kripke Prop State
exampleModel = Kripke [0,1] statespace transition owner val
```

The `transition` relation over the `statespace` describes the possible transitions resulting from agents' actions. In addition to these components we also need to assign the `owner` of each transition step. The proposition symbols and their valuation map complete the definition of the model.

Regarding the type of the propositions, `Prop`, we would like to talk about whether a particular agent is found on the east/west edge of the pavement, or in the north/south "lane". The type of the states, `State`, must be able to code the current agent and the positions of both the agents. We can represent this by three integers.

```
data Property = N | S | E | W | T deriving Eq

type Prop  = (Property, Int)
type State = (Int, Int, Int)
```

A state is thus a tuple consisting of three integers. The first indicates the position of agent `0`, the second the position of agent `1`, and the last indicates which agent's turn it is. The sorted list of possible states of the model can be described by Haskell's list comprehension syntax succinctly:

```
statespace :: [State]
statespace = [ (p0, p1, i) | p0 ← [0..9],
                             p1 ← [0..9],
                             p0 /= p1, i ← [0,1] ]
```

The valuation function selects the states from `statespace` in which the indicated agent is in one of the appropriate positions.

```
project :: Int → State → Int
project 0 (p0, _, _) = p0
project 1 (_, p1, _) = p1

val :: Prop → [State]
val (N, ag) = filter (even ∘ (project ag)) statespace
val (S, ag) = filter (odd ∘ (project ag)) statespace
val (W, ag) = filter ((`elem` [0,1]) ∘ (project ag)) statespace
val (E, ag) = filter ((`elem` [8,9]) ∘ (project ag)) statespace
val (T, ag) = filter (λ(_,_,i) → i == ag) statespace
```

The `transition` relation indicates which transitions are available to the agents, or simply what 'moves' agents can perform.

```
transition = build [((p0,p1,i), (p0',p1',1-i)) |
                    (p0,p1,i) ← statespace,
                    p0'       ← possibleSteps (i == 0) p0,
                    p1'       ← possibleSteps (i == 1) p1,
                    (p0',p1',1-i) `elem` statespace]
```

`possibleSteps` is a helper function for `transition`, and says that agents can 'stand', 'move' west, east, and so on. If it's not the agent's turn (first argument is `False`) the only possible move is standing.

```
possibleSteps :: Bool → Int → [Int]
possibleSteps False p = [p]
possibleSteps True p = [p-2, p, p+2]++(if even p then [p+1]
                                                 else [p-1])
```

Normative systems are represented in the same way as the transition relation, through the `FODBR` type. The following relation `illegal` specifies a normative system for the example: any agent must not place himself directly in front of the other agent, and if an agent is able to move in the general direction required for the satisfaction of the goal, he must do so.

```
illegal :: FODBR State State
illegal = build [ ((p0,p1,i),(p0',p1',1-i)) |
                  (p0,p1,i)   ← statespace,
                  (p0',p1',_) ← (find1 (fst transition) (p0,p1,i)),
                  p1' == p0' + 2 || if i == 0
                                    then (p0' < 8 && p0 ≥ p0')
                                    else (p1' > 1 &&  p1 ≤ p1')]

owner :: (State,State) → Int
owner ((_,_,i),_) = i
```

Finally, formulas are represented by the structures `Formula` and `Coalition`:

```
data (Eq p) => Formula p = Prop p           | Neg  (Formula p)
             | Disj (Formula p) (Formula p) | Conj (Formula p) (Formula p)
             | EX   (Formula p)             | EF   (Formula p)
             | EG   (Formula p)             | EU   (Formula p) (Formula p)
             | CD   Coalition   (Formula p) | CS   Coalition   (Formula p)
               deriving (Show)

data Coalition = Subseteq [Int]             | Supseteq [Int]
               | Eq [Int]                   | GEQ Int
               | CNeg Coalition             | CDisj Coalition Coalition
                 deriving (Show)

ag,af :: (Eq p) => (Formula p) -> (Formula p)
ag f = Neg (EF (Neg f))
af f = Neg (EG (Neg f))
```

A formula that should be true only in the initial state in the example:

```
initF :: Formula Prop
initF = Conj (Conj (Prop (T, 0)) (Conj (Prop (N, 0)) (Prop (W, 0))))
             (Conj (Prop (S, 1)) (Prop (E, 1)))
```

And now an *objective* formula: agent `0` is east, and `1` is west:

```
goalF :: Formula Prop
goalF = Conj (Prop (E, 0)) (Prop (W, 1))
```

We now have all three arguments for the `check` function:

```
*Ex01> check exampleModel illegal (Conj initF (af goalF))
[]
*Ex01> check exampleModel illegal (CD (GEQ 0) (Conj initF (af goalF)))
[(0,9,0)]
```

We can see that there is no state satisfying both `initF` and `af goalF`, but there exists a coalition which can ensure that we eventually end up satisfying the goal.

4 SPECIFYING AND MODEL CHECKING A MORE COMPLEX SYSTEM

A more elaborate example from [8] describes four researches attending a conference who need to share a limited amount of resources. The model in the example has 62500 states and 470596 transitions. We present the essential parts of this example and use it to demonstrate the flexibility in describing more elaborate models in Haskell/NorMC, and that NorMC can model check also larger models. Due to lack of space some details must be left out, but we focus on the essential parts.

The states are tuples $s = \langle O_a, O_b, O_c, O_d, i \rangle$ where for each $i \in \{a, b, c, d\} = A$, O_i represents the resources owned by agent i. The resources they need

to share are: (i) a printer (the set R_1), (ii) two scanners (the set R_2), and (iii) three computers (the set R_3). The agents do not have the same needs. Agent a needs the printer and a computer, agent b needs a scanner and the printer, agent c needs a scanner and a computer, and agent d only needs the printer. Agents' actions are turn-based. There is no distribution of resources that allows all agents to own the resources they need simultaneously, but there are limitations on the permitted transitions which guarantee that each agent will eventually own all the resources needed.

The first restriction is the implementation of a normative system which sets out some basic rules of behaviour. Since the example is rather elaborate, we have to omit much of the formal definition here and refer to [8] for details. Informal description of the normative system η_0 [8]:

> [...] no agent (i) owns two resources of the same type at the same time, (ii) takes possession of a resource that he does not need, (iii) takes possession of two new resources simultaneously, and (iv) fails to take possession of some useful resource if it is available when it is his turn [...]

We implement an equivalent model where states are represented by seven integers. The first six of these integers represent the owner of, respectively, (1) the printer, (2) scanner_1, (3) scanner_2, (4) computer_1, (5) computer_2 and (6) computer_3. Each of these can take on a value 0..4 where 0 represents a *free* resource and a non-zero number indicates the owner of this resource. The seventh integer represent whose turn it is and has a value 1..4.

```
type State = (Int, Int, Int, Int, Int, Int, Int)
```

The set of states is defined as follows.

```
statespace :: [State]
statespace = sort
             [(p, s1, s2, c1, c2, c3, a) |
                   p  ← [0..4], s1 ← [0..4], s2 ← [0..4],
                   c1 ← [0..4], c2 ← [0..4], c3 ← [0..4],
                                             a  ← [1..4] ]
```

For the transition relation, if the current agent `a` owns a given resource r (`a` $==$ r), or nobody owns that resource (r $==$`0`), then the current agent can keep (resp. grab) or release (resp. ignore) it, otherwise that resource will not change owner.

```
transition :: FODBR State State
transition = build [((p , s1 , s2 , c1 , c2 , c3 , a),
                     (p', s1', s2', c1', c2', c3', 1 + (a `mod` 4))) |
                    (p,s1,s2,c1,c2,c3,a) ← statespace,
                    p'  ← if (p  == a || p  == 0)
                                         then [0, a] else [p ],
                    s1' ← if (s1 == a || s1 == 0)
                                         then [0, a] else [s1],
                    s2' ← if (s2 == a || s2 == 0)
```

```
                                then [0, a] else [s2],
        c1' ← if (c1 == a || c1 == 0)
                                then [0, a] else [c1],
        c2' ← if (c2 == a || c2 == 0)
                                then [0, a] else [c2],
        c3' ← if (c3 == a || c3 == 0)
                                then [0, a] else [c3] ]
```

We label each transition with its owner. This is a simple projection.

```
owner :: (State, State) → Int
owner ((_,_,_,_,_,_,i), _) = i
```

As in the previous example, we model a proposition symbol by an (agent) index and a keyword. We define the type `Resource` to denote the various resources and give a function which projects from a given state the owner of the given resource.

```
data Resource = Pr | S1 | S2 | C1 | C2 | C3 deriving Eq

project :: Resource → State → Int
project Pr (pr,_,_,_,_,_,_) = pr
project S1 (_,s1,_,_,_,_,_) = s1
-- and so on...
```

The valuation function is now simply defined by removing the states in which given agent does not own the resource in question.

```
type Proposition = (Resource, Int)

val :: Proposition → [State]
val (res, ag) = filter ((ag == ) ∘ (project res)) statespace
```

The normative system η_0 is implemented as the union of four separate relations (see the description of η_0).These are all constructed in a similar way, i.e. restricting the transition relation appropriately, and we show only `component3` which does not allow an agent to grab two resources at the same time.

```
component3 :: FODBR State State
component3 = restrict transition
        (λ(pr,s1,s2,c1,c2,c3,a) (pr', s1', s2', c1', c2', c3', a') →
           (foldr (+) 0 $ zipWith (λx y → if (x /= a) && (y == a)
              then 1 else 0)
                            [pr , s1 , s2 , c1 , c2 , c3 ]
                            [pr', s1', s2', c1', c2', c3']) > 1)

eta_0 :: FODBR State State
eta_0 = component1 `union` component2 `union` component3 `union`
    component4
```

Next we define the normative system η_1: if an agent owns all his useful resources simultaneously (he is 'happy'), he will make them available in the next turn.

```
stHappy :: Int → State → Bool
stHappy 1 (pr, _,  _,  c1, c2, c3, _) =
                  (pr == 1 && (c1 == 1 || c2 == 1 || c3 == 1))
-- and similarly for stHappy 2, 3 and 4, defining the states where
   agents are happy

ownsSomething :: Int → (State → Bool)
ownsSomething n =(pr, s1, s2, c1, c2, c3, a) → (pr == n || s1 == n ||
                                                s2 == n || c1 == n ||
                                                c2 == n || c3 == n)

eta_1 :: FODBR State State
eta_1 = restrict transition
        (λs s' → (stHappy (owner (s,s')) s) && (ownsSomething (owner
           (s,s')) s'))
```

For convenience, we define two example models, one with no system implemented on it (K_0), and the second one with η_0 implemented on it ($K_1 = K_0 \dagger \eta_0$):

```
exampleModel :: Kripke Proposition State
exampleModel = Kripke [1,2,3,4] statespace transition owner val

exampleModel' :: Kripke Proposition State
exampleModel' = ir exampleModel eta_0 [1,2,3,4]
```

Our first objective is that it is always the case that every agent will eventually become happy: $\varphi_1 = \mathsf{A}\Box(\bigwedge_{i \in Ag} \mathsf{A}\Diamond happy(i))$, where the proposition $happy(i)$ (code: `fHappy i`) is true iff i is 'happy'.

```
phi_1 :: Formula Proposition
phi_1 =  ag (Conj (af (fHappy 1))(Conj(af (fHappy 2))(Conj(af (fHappy
   3))(af (fHappy 4)))))
```

This concludes the implementation of the example model from [8]. We now proceed with model checking. First we check whether `eta_0` is effective.

```
*Ex02> check exampleModel' eta_1 phi_1
[]
```

The model checker outputs an empty set, indicating there are no states satisfying `phi_1` if no agents are required to comply with `eta_1`. In [8] it is claimed that compliance of coalition $\{1,2,3\}$ to `eta_1` is sufficient for the objective to hold (assuming that all agents comply with η_0). Let us check:

```
*Ex02> check exampleModel' eta_1 (CS (Supseteq [1,2,3]) phi_1)
[]
```

Surprisingly, the model checker tells us that the formula is not true in any state. Using the model checker we can produce a trace:

```
[(0,0,0,0,0,0,1),(0,0,0,0,0,1,2),(2,0,0,0,0,1,3),(2,0,0,0,3,1,4),
 (2,0,4,0,3,1,1),(2,0,4,0,3,0,2),(0,2,4,0,3,0,3),(0,2,4,0,0,0,4),
 (0,2,0,0,0,0,1),(0,2,0,0,0,1,2),(2,0,0,0,0,1,3)]
```

In steps 2 and 10 we end up in the same state. We have a loop where all agents comply with both η_0 and η_1. Only one agent (4) is ever 'happy' in this loop. So, there is a path π in $\Pi(s_0)$ along which it is not the case that every agent will eventually be 'happy'. Thus, there is an error in the example in [8]. This error was not obvious, and this illustrates the benefit of software tools even for "toy" examples.

The problem with the example from [8] is that η_1 as stated is too weak; as seen above it allows agents to simultaneously grab some resources and release others in an endless loop without ever becoming 'happy'. We now introduce an additional condition that tells the agents to not release any resources they posses until they are 'happy':

```
dont_release :: FODBR State State
dont_release = restrict transition
             (λs@(pr,s1,s2,c1,c2,c3,a) (pr',s1',s2',c1',c2',c3',_) →
                 (not $ stHappy a s) &&
                 (((pr == a) && (pr' /= a)) ||
                  ((s1 == a) && (s1' /= a)) ||
                  ((s2 == a) && (s2' /= a)) ||
                  ((c1 == a) && (c1' /= a)) ||
                  ((c2 == a) && (c2' /= a)) ||
                  ((c3 == a) && (c3' /= a))))
```

Modify the normative system `eta_1` to include the new condition:

```
eta_1' :: FODBR State State
eta_1' = eta_1 `union` dont_release
```

Now $\{1,2,3\}$ is a sufficient[1] coalition (the only minimal sufficient coalition):

```
*Ex02> let is = ((0,0,0,0,0,0,1)::State)
*Ex02> is `elem` check exampleModel' eta_1' (CS (Supseteq [1,2,3]) phi_1)
True
```

5 IMPLEMENTATION

We now discuss the design and implementation of NorMC. As mentioned, the function `check` is the core of the model checker. It implements an extension of a standard model checking algorithm for CTL formulas, as described in [32], with additional clauses for coalition predicates. `check` takes a model, normative system and formula as arguments and returns the set of states in which the given formula is satisfied. The function is

1 C is sufficient for φ in the context of K and η iff $\forall C' \subseteq A : (C \subseteq C') \Rightarrow [K \dagger (\eta \restriction C') \models \varphi]$. Note that $K \dagger (\eta \restriction C) \models \varphi$ does not in general imply that $K \dagger (\eta \restriction C') \models \varphi$ when $C \subseteq C'$ [8].

defined recursively, and the clauses for propositional variables and path-state-quantifiers are defined as in standard CTL algorithms.

For the coalition predicates we need some semantic update functions: `nsimplement` implements a normative system on a Kripke structure, `nsrestrict` implements the $\restriction$ operator, and `ir` combines the two, implementing a given normative system restricted to the given coalition predicate:

```
nsimplement :: (Ord s, Eq p) ⇒ (Kripke p s) → (FODBR s s)
                                             → (Kripke p s)
nsimplement mod sys = mod { tr = (tr mod) `minus` sys }

nsrestrict :: (Ord s, Eq p) ⇒ (Kripke p s) → (FODBR s s)
                                            → [Int] → (FODBR s s)
nsrestrict mod sys coa = restrict sys (λs s' → ((owner mod (s,s'))
                                                `elem` coa))

ir :: (Ord s, Eq p) ⇒ (Kripke p s) → (FODBR s s) → [Int]
                                                 → (Kripke p s)
ir mod sys coa = nsimplement mod (nsrestrict mod sys coa)
```

We now describe some minor optimisation tricks used to make NORMC usable for non-trivial examples.

Model checking NCCTL involves quantification over coalitions. As discussed, for each subformula φ the algorithm computes the set of satisfying states (call it $sat(\varphi)$). By the semantics of NCCTL, we have that $sat([P]\varphi) = \{s \in S : \forall C \subseteq A(C \models_{cp} P \Rightarrow K \dagger (\eta \restriction C), \eta, s \models \varphi)$. A naive implementation of this involves testing every coalition against the predicate, once for each state. However, it is easy to see that the quantifier can be moved out (and similarly for $\langle P \rangle \varphi$):

LEMMA 5.1: *$sat([P]\varphi) = \bigcup_{C \models_{cp} P} \{s \in S : K \dagger (\eta \restriction C), \eta, s \models \varphi)$ and $sat(\langle P \rangle \varphi) = \bigcap_{C \models_{cp} P} \{s \in S : K \dagger (\eta \restriction C), \eta, s \models \varphi)$.*

Thus we only have to test the predicate once for each coalition, and save a considerable amount of time in practice. This is made use of as follows:

```
check mod sys (CD c f) = foldl' nubunion [] $
                         map (λmod → (check mod sys f)) $
                             map (ir mod sys) $ coasGivenCP mod c

check mod sys (CS c f) = foldl' nubisect (states mod) $
                         map (λmod → (check mod sys f)) $
                             map (ir mod sys) $ coasGivenCP mod c
```

Much of the core functionality of NORMC is found in the handling of the transition relation, and performing a model update operation is performing an appropriate restriction on this relation. We only mention the implementation of relation handling briefly here, due to lack of space and to keep focus on more high-level considerations. Our relation handling implementation, `FODBR` (Finite Ordered Domain Binary Relation), is an efficient representation of a binary relation by two binary search

trees. We require that the domain is finite and consists of elements with an ordering, and we represent the binary relation by two multifunctions: $src : \mathcal{D} \to \wp(\mathcal{D}')$ and $trg : \mathcal{D} \to \wp(\mathcal{D}')$. Both these functions are stored in a binary search tree where each node contains a value of the type $(\mathcal{D}, [\mathcal{D}'])$ ('key' and 'value set', respectively). The functionality provided by `FODBR` requires that argument lists are sorted and contain no duplicate elements, and guarantees that any returned lists also satisfy these properties. This allows us to implement the usual set theoretic operations of union, intersection and difference efficiently.

6 CONCLUSIONS

In this paper we have described NorMC, a prototype model checker for Norm Compliance CTL (NCCTL). NCCTL extends CTL with a family of modalities with update semantics. We have also aimed to demonstrate that the Haskell programming language is a natural and useful alternative for model checking; in particular that the full power of Haskell makes it easy to describe arbitrary state-transition models in a natural way. While NorMC has not been optimised for industrial use, we have seen that it is efficient enough to be used on interesting and non-trivial examples. If higher efficiency is needed, there are two natural options. The first is to extend NorMC with standard symbolic model checking techniques such as binary decision diagrams (BDDs) and partial order reduction. The other is to change NorMC into a front-end for an efficient existing CTL model checker. For future work, we are currently implementing a graphical front-end to NorMC.

ACKNOWLEDGEMENTS Piotr Kaźmierczak and Thomas Ågotnes' work has been supported by the Research Council of Norway project 194521 (FORMGRID). The authors also thank Paul Simon Svanberg for helpful comments.

BIBLIOGRAPHY

[1] T. Ågotnes. Action and Knowledge in Alternating-Time Temporal Logic. *Synthese*, 149(2):375–407, 2006.

[2] T. Ågotnes and M. Wooldridge. Optimal Social Laws. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 667–674, Toronto, Canada, May 10–14 2010.

[3] T. Ågotnes, V. Goranko, and W. Jamroga. Alternating-time temporal logics with irrevocable strategies. In *Proceedings of the 11th conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, pages 15–24, New York, New York, USA, 2007. ACM.

[4] T. Ågotnes, W. van der Hoek, J. A. Rodríguez-Aguilar, C. Sierra, and M. Wooldridge. On the Logic of Normative Systems. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 07)*, pages 1175–1180, 2007.

[5] T. Ågotnes, W. van der Hoek, and M. Wooldridge. Normative system games. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007)*, pages 1–8, Honolulu, USA, May 14–18 2007.

[6] T. Ågotnes, W. van der Hoek, and M. Wooldridge. Quantified coalition logic. *Synthese (Special Section on Knowledge, Rationality and Action)*, 165(2):269–294, 2008.

[7] T. Ågotnes, W. van der Hoek, M. Tennenholtz, and M. Wooldridge. Power in normative systems. In *Proceedings of 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, pages 145–152, Budapest, Hungary, May 10–15 2009.

[8] T. Ågotnes, W. van der Hoek, and M. Wooldridge. Robust normative systems and a logic of norm compliance. *Logic Journal of the IGPL*, 18(1):4–30, 2009.

[9] T. Ågotnes, W. van der Hoek, and M. Wooldridge. Conservative Social Laws. In L. D. Raedt, C. Bessiere, D. Dubois, P. Doherty, P. Frasconi, F. Heintz, and P. Lucas, editors, *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI 2012)*, pages 49–54, Montpellier, France, August 2012.

[10] N. Alechina, B. Logan, H. Nguyen, and A. Rakib. Resource-bounded alternating-time temporal logic. In *Proceedings of AAMAS'10*, pages 481–488, 2010.

[11] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 100–109. IEEE Computer Society Press, 1997.

[12] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. *Journal of the ACM*, 49(5):672–713, 2002.

[13] R. Axelrod. *The Evolution of Cooperation*. Basic Books, 1984.

[14] Y. Bachrach and J. S. Rosenschein. Coalitional skill games. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 1023–1030. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[15] J. W. Backus, F. L. Bauer, J. Green, C. Katz, J. McCarthy, A. J. Perlis, H. Rutishauser, K. Samelson, B. Vauquois, J. H. Wegstein, A. van Wijngaarden, and M. Woodger. Report on the Algorithmic Language ALGOL 60. *Communications of the ACM*, 3(5):299–314, 1960.

[16] J. F. Banzhaf. Weighted voting doesn't work: A mathematical analysis. *Rutgers Law Review*, 19:317–343, 1965.

[17] M. Bauland, M. Mundhenk, T. Schneider, H. Schnoor, I. Schnoor, and H. Vollmer. The Tractability of Model-checking for LTL: The Good, the Bad, and the Ugly Fragments. In *Proceedings of the 5th Workshop on Methods for Modalities (M4M5 2007)*, volume 231 of *Electronic Notes in Theoretical Computer Science*, pages 277–292, 2009.

[18] N. Belnap, M. Perloff, and M. Xu. *Facing the Future: Agents and Choices in Our Indeterminist World*. Oxford University Press, 2001.

[19] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

[20] M. Blonski. Characterization of pure strategy equilibria in finite anonymous games. *Journal of Mathematical Economics*, 34:225–233, 2000.

[21] G. Boella, L. van der Torre, and H. Verhagen. Introduction to normative multiagent systems. *Computational & Mathematical Organization Theory*, 12(2–3):71–79, 2006.

[22] F. Brandt, F. Fischer, and M. Holzer. Symmetries and the complexity of pure nash equilibrium. *Journal of Computer and System Sciences*, 75 (3):163–177, 2009.

[23] F. Brandt, V. Conitzer, and U. Endriss. Computational Social Choice. In G. Weiss, editor, *Multiagent Systems*, pages 213–284. MIT Press, 2014.

[24] T. Brihaye, A. D. C. Lopes, F. Laroussinie, and N. Markey. ATL with strategy contexts and bounded memory. In *Proceedings of LFCS*, volume 5407 of *LNCS*, pages 92–106, 2009.

[25] S. J. Buckley. Fast Motion Planning for Multiple Moving Robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 322–326, Scottsdale, AZ, 1989.

[26] N. Bulling and J. Dix. Modelling and verifying coalitions using argumentation and ATL. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 14(46):45–73, 2010.

[27] N. Bulling and B. Farwer. On the (un-)decidability of model checking resource-bounded agents. In *Proceedings of ECAI*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 567–572. IOS Press, 2010.

[28] G. Chalkiadakis, E. Elkind, and M. Wooldridge. *Computational Aspects of Cooperative Game Theory*. Number 16 in Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2012.

[29] K. Chatterjee, T. A. Henzinger, and N. Piterman. Strategy logic. In *Proceedings of CONCUR*, pages 59–73, 2007.

[30] A. K. Chopra and M. P. Singh. Agent Communication. In G. Weiss, editor, *Multiagent Systems*, pages 101–141. MIT Press, 2013.

[31] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV 2: An OpenSource Tool for Symbolic Model Checking. In *Proceedings of International Conference on Computer-Aided Verification (CAV 2002)*, volume 2404 of *LNCS*, 2002.

[32] E. M. Clark, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[33] E. M. Clarke and E. A. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In D. Kozen, editor, *Logic of Programs Workshop*, volume 131 of *LNCS*. Springer-Verlag, 1981.

[34] M. Davis and H. Putnam. A Computing Procedure for Quantification Theory. *Journal of the ACM*, 7(3):201–215, 1960.

[35] R. Davis and R. G. Smith. Negotiation as a Metaphor for Distributed Problem Solving. *Artificial Intelligence*, 20(1):63–109, 1983.

[36] P. Dellunde. On the multimodal logic of normative systems. In *Proceedings of COIN'07*, pages 261–274, 2008.

[37] R. Diaconu and C. Dima. Model-checking alternating-time temporal logic with strategies based on common knowledge is undecidable. *Applied Artificial Intelligence*, 26(4):331–348, 2012.

[38] C. Dima and F. L. Tiplea. Model-checking ATL under imperfect information and perfect recall semantics is undecidable. *CoRR*, abs/1102.4225, 2011.

[39] C. Dima, C. Enea, and D. Guelev. Model-checking an alternating-time temporal logic with knowledge, imperfect information, perfect recall and communicating coalitions. In *Proceedings of the 1st International Symposium on Games, Automata, Logics and Formal Verification (GandALF 2010)*, Electronic Proceedings in Theoretical Computer Science, pages 103–117, June 2010.

[40] K. Doets and J. van Eijck. *The Haskell Road to Logic, Maths and Programming*, volume 4 of *Texts in Computing*. College Publications, 2004.

[41] P. E. Dunne, W. van der Hoek, S. Kraus, and M. Wooldridge. Cooperative Boolean Games. In *Proceedings of 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pages 1015–1022, Estoril, Portugal, May 12–16 2008.

[42] S. Dyrkolbotn and P. Kaźmierczak. Playing with Norms: Tractability of Normative Systems for Homogeneous Game Structures. In A. Lomuscio, P. Scerri, A. Bazzan, and M. Huhns, editors, *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014)*, pages 125–132, Paris, France, May 5-9 2014.

[43] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science Volume B: Formal Models and Semantics*. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990.

[44] M. Erdmann and T. Lozano-Pérez. On Multiple Moving Objects. *Algorithmica*, 2:477–521, 1987.

[45] D. Fitoussi and M. Tennenholtz. Choosing social laws for multi-agent systems: Minimality and simplicity. *Artificial Intelligence*, 119: 61–101, 2000.

[46] M. Fitting. *First-order logic and automated theorem proving*. Springer, 1990.

[47] D. B. Gillies. Solutions to general non-zero-sum games. In A. W. Tucker and R. D. Luce, editors, *Contributions to the Theory of Games IV*, number 40 in Annals of Mathematics Studies, pages 47–85. Princeton University Press, Princeton, 1959.

[48] D. Guelev and C. Dima. Model-checking strategic ability and knowledge of the past of communicating coalitions. In *Proceedings of DALT 2008*, volume 5397 of *Lecture Notes in Computer Science*, pages 75–90. Springer, 2008.

[49] D. Guelev, C. Dima, and C. Enea. An alternating-time temporal logic with knowledge, perfect recall and past: axiomatisation and model-checking. *Journal of Applied Non-Classical Logics*, 21(1):93–131, 2011.

[50] P. Hall. On representatives of subsets. *Journal of the London Mathematical Society*, s1-10(1):26–30, 1935.

[51] J. Hintikka. *Knowledge and Belief: an Introduction to the Logic of the Two Notions*. Ithaca: Cornell University Press, 1962.

[52] G. Holzmann. The Spin model checker. *IEEE Transactions on Software Engineering*, 23:279–295, 1997.

[53] J. F. Horty. *Agency and Deontic Logic*. Oxford University Press, 2001.

[54] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, second edition, 2009.

[55] W. Jamroga. A temporal logic for stochastic multi-agent systems. In *Proceedings of PRIMA'08*, volume 5357 of *LNCS*, pages 239–250, 2008.

[56] W. Jamroga and T. Ågotnes. Constructive knowledge: what agents can achieve under imperfect information. *Journal of Applied Non-Classical Logics*, 17(4):423–475, Dec. 2007. ISSN 11663081. doi: 10.3166/jancl.17.423-475.

[57] W. Jamroga and J. Dix. Do agents make model checking explode (computationally)? In M. Pechoucek, P. Petta, and L. Z. Varga, editors, *Multi-Agent Systems and Applications IV (LNAI Volume 3690)*, 2005.

[58] W. Jamroga and A. Murano. On module checking and strategies. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014)*, pages 701–708, 2014.

[59] W. Jamroga and W. van der Hoek. Agents that Know How to Play. *Fundamenta Informaticae*, 63(2-3):185–219, 2004.

[60] S. P. Jones. *Haskell 98 language and libraries: the Revised Report*. Cambridge University Press, 2003.

[61] P. Kaźmierczak. Compliance Games. In *Proceedings of SOCREAL 2013: 3rd International Workshop on Philosophy and Ethics of Social Reality 2013*, pages 126–131, Hokkaido, Japan, October 2013. Hokkaido University Collection of Scholarly and Academic Papers.

[62] W. Kneale and M. Kneale. *The Development of Logic*. Oxford University Press, 1962.

[63] S. Kraus and J. Wilkenfeld. The Function of Time in Cooperative Negotiations. In *Proceedings of AAAI-91*, pages 179–184, Anaheim, CA, 1991.

[64] S. Kripke. A Completeness Theorem in Modal Logic. *Journal of Symbolic Logic*, 24:1–14, 1959.

[65] S. Kripke. Semantical Considerations on Modal Logic. *Acta Philosophica Fennica*, 16:83–94, 1963.

[66] S. Kripke. Semantic analysis of modal logic I, normal propositional calculi. *Zeitschrift für mathemathische Logik und Grundlagen der Mathematik*, 9:67–96, 1963.

[67] E. J. Lemmon and D. Scott. *An Introduction to Modal Logic*. Oxford: Blackwell, 1977.

[68] K. Leyton-Brown and Y. Shoham. *Essentials of Game Theory*. Morgan & Claypool, 2008.

[69] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A model checker for the verification of multi-agent systems. In *Proceedings of CAV'09*, volume 5643 of *LNCS*, pages 682–688, 2009.

[70] Y. Moses and M. Tennenholtz. Artificial Social Systems. Technical Report CS90-12, Weizmann Institute, Rehovot, Israel, 1990.

[71] Y. Moses and M. Tennenholtz. On computational aspects of artificial social systems. In *Proceedings of Workshop on Distributed Artificial Intelligence*, 1992.

[72] J. F. Nash. Equilibrium Points in *N*-Person Games. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 36, pages 48–49, 1950.

[73] N. Nisan. Introduction to mechanism design (for computer scientists). In N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*, pages 209–242. Cambridge University Press, 2007.

[74] J. O'Donnell, C. Hall, and R. Page. *Discrete Mathematics Using a Computer*. Springer, second edition, 2006.

[75] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[76] M. Pauly. *Logic for Social Software*. ILLC dissertation series 2001-10, University of Amsterdam, 2001.

[77] M. Pauly. A Modal Logic for Coalitional Power in Games. *Journal of Logic and Computation*, 12(1):149–166, 2002.

[78] T. Pedersen and S. Dyrkolbotn. Agents homogeneous: A procedurally anonymous semantics characterizing the homogeneous fragment of ATL. In *Proceedings of PRIMA 2013*, LNAI, 2013.

[79] T. Pedersen, S. Dyrkolbotn, and P. Kaźmierczak. Big, but not unruly: Tractable norms for anonymous game structures. arXiv:1405.6899, 2013.

[80] T. Pedersen, S. Dyrkolbotn, P. Kaźmierczak, and E. Parmann. Concurrent Game Structures with Roles. In *Proceedings of the 1st International Workshop on Strategic Reasoning*, volume 112 of *EPTCS*, pages 61–69, Rome, Italy, 2013. Open Publishing Association.

[81] J. Pilecki, M. Bednarczyk, and W. Jamroga. Synthesis and verification of uniform strategies for multi-agent systems. In *Proceedings of the 15th Workshop on Computational Logic in Multi-agent Systems (CLIMA XV)*, volume 8624 of *LNCS*, pages 166–182, 2014.

[82] A. Pnuelli. The Temporal Logic of Programs. In *Proceedings of the 18th International Symposium on Foundations of Computer Science*, pages 46–57. IEEE Computer Society Press, 1977.

[83] A. N. Prior. *Time and Modality*. Oxford University Press, 1957.

[84] A. N. Prior. *Past, Present and Future*. Oxford University Press, 1967.

[85] A. N. Prior. *Papers on time and tense*. Oxford University Press, 1968.

[86] J. P. Quielle and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *Proceedings of the Fifth International Symposium on Programming*, 1981.

[87] J. S. Rosenschein and M. R. Genesereth. Deals Among Rational Agents. In *Proceedings of IJCAI-85*, pages 91–99, Los Angeles, CA, 1985.

[88] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 1995.

[89] S. J. Russell and D. Subramanian. Provably bounded-optimal agents. *Journal of Artificial Intelligence Research*, 2:579–609, 1995.

[90] H. Schnoor. Strategic planning for probabilistic games with incomplete information. In *Proceedings of AAMAS'10*, pages 1057–1064, 2010.

[91] P.-Y. Schobbens. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, 85(2):82–93, 2004.

[92] L. S. Shapley. A value for n-person games. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games*, volume II, pages 307–317. Princeton University Press, 1953.

[93] L. S. Shapley and M. Shubik. A Method for Evaluating the Distribution of Power in a Committee System. *American Political Science Review*, 48:787–792, 1954.

[94] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.

[95] Y. Shoham and M. Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In *Proceedings of the tenth national conference on Artificial intelligence*, AAAI'92, pages 276–281. AAAI Press, 1992.

[96] Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: off-line design. *Artificial Intelligence*, 73:231–252, 1995.

[97] A. P. Sistla and E. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32:733–749, 1985.

[98] W. van der Hoek and M. Wooldridge. Time, knowledge, and cooperation: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75(1):125–157, 2003.

[99] W. van der Hoek and M. Wooldridge. Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75:125–157, 2003. ISSN 0039-3215. 10.1023/A:1026185103185.

[100] W. van der Hoek and M. Wooldridge. Logics for multiagent systems. In G. Weiss, editor, *Multiagent Systems*, pages 761–810. MIT Press, Cambridge, MA, USA, 2013.

[101] W. van der Hoek, W. Jamroga, and M. Wooldridge. A logic for strategic reasoning. In *Proceedings of AAMAS'05*, pages 157–164, 2005.

[102] W. van der Hoek, M. Roberts, and M. Wooldridge. Social laws in alternating time: effectiveness, feasibility, and synthesis. *Synthese*, 156(1):1–19, 2006.

[103] J. van Eijck. Dynamic epistemic modelling. Technical report, CWI and ILLC, Amsterdam, Uil-OTS, Utrecht, http://homepages.cwi.nl/~jve/demo/, Summer 2005.

[104] S. van Otterloo and G. Jonker. On Epistemic Logic and its applications. In *Electronic Notes in Theoretical Computer Science*, volume 126 of *Proceedings of LCMAS'04*, pages 77–92, 2004.

[105] S. van Otterloo, W. van der Hoek, and M. Wooldridge. Preferences in game logics. In *Proceedings of AAMAS'04*, pages 152–159, 2004.

[106] D. Walther, W. van der Hoek, and M. Wooldridge. Alternating-time temporal logic with explicit strategies. In D. Samet, editor, *Proceedings of TARK XI*, pages 269–278, 2007.

[107] G. Weiss, editor. *Multiagent Systems*. MIT Press, Cambridge, MA, USA, 2013.

[108] M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons Ltd, second edition, 2009.

[109] M. Wooldridge. Intelligent Agents. In G. Weiss, editor, *Multiagent Systems*, pages 3–50. MIT Press, Cambridge, MA, USA, 2013.

[110] M. Wooldridge and P. E. Dunne. On the Computational Complexity of Qualitative Coalitional Games. *Artificial Intelligence*, 158, 2004.

[111] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Final Version as of March 26, 2015 (`classicthesis` version 0.1).