Logics for Modelling and Verifying Normative Multi-agent Systems

Max Knobbout

This research has been supported by the Netherlands Organisation for Scientific Research (NWO).

SIKS Dissertation Series No. 2016-12
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

Published by Proefschriftomslag.nl
Printed by Ridderprint, Ridderkerk
Cover design by Esther Ris

ISBN 978-94-92332-08-0

Logics for Modelling and Verifying Normative Multi-Agent Systems

Logica's voor het Modelleren en Verifiëren van Normatieve Multi-Agent Systemen

(met een samenvatting in het Nederlands)

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Universiteit Utrecht op gezag van de rector magnificus, prof. dr. G.J. van der Zwaan, ingevolge het besluit van het college voor promoties in het openbaar te verdedigen op woensdag 6 april 2016 des middags te 12.45 uur

door

Max Knobbout

geboren op 17 februari 1987 te Ede

Promotor: Prof. dr. J.-J. Ch. Meyer
Copromotor: Dr. M.M. Dastani

Contents

Chapter 1

Introduction

A multi-agent system is a computerized system that is composed of multiple interacting agents within an environment. Such an agent can be a software program, but could equally well be a human actor. These systems are making their way into our everyday life, for example electronic marketplaces where agents can buy and sell goods, or smart grid energy systems where agents can produce or consume energy based on supply and demand. These systems are typically distributed and decentralized, and the participating agents may have diverging information, diverging interests, or both. The internals and architecture of these participating agents may be unknown to us, which disables us to make any strong assumption on the possible behaviour that these agents may exhibit. Moreover, these systems are generally composed or designed with a specific goal in mind, and depending on the behaviour of the system these goals may, or may not, be achieved. Hence there is a need to regulate and control the behaviour of the system and the participating agents. In order to achieve this, norms have been proposed to regulate, coordinate, and control the behaviour of the agents and system, leading to the field of research called normative multi-agent systems. In this thesis we study enforcement norms, which are norms which can be violated, and which can lead to sanctions when a violation is detected. However, it can be the case that the norms are not well-designed, and that there exists ways for an agent to exploit the system. This may for example lead to dangerous situations within the system, or high costs for the designer. As these normative multi-agent systems can grow in size and scale, formal methods and tools are needed in order to prove whether such a system works as intended. This thesis aims to design and analyse formal logic-based frameworks in which normative multi-agent systems can be modelled and verified.

1.1 Motivation

This section will try to motivate why the formal logic-based analysis of normative multi-agent systems is important. Particularly, we will concern ourselves with the following questions

1. Why is the research field of normative multi-agent systems important?
2. Why do we need to concern ourselves with formal logic-based frameworks to model and verify these normative multi-agent systems?

In order to answer the first question, it is crucial to realize that a multi-agent system is generally composed or designed with a specific goal and mind. For example, an electronic marketplace consisting of several buyer and seller agents can be used to perform safe transactions between currency and goods. However, agents within such a system can have their own interests and goals, and these goals may be unaligned with the overall objective of such a system. Hence there is a need to regulate and control the behaviour of the system and the participating agents. Since we may not have access to the internal architecture of an agent, we may not know whether an agent is benevolent or not, so typically such information can only be generated retroactively. The crucial notion here is *control*: How can we control a multi-agent system in such a way to guarantee that the overall objectives of the system are met? Control is of crucial importance whenever defection is costly. In an electronic marketplace, defection can imply a loss of money, and in a smart road system defection can even imply physical injury. In literature we can identify at least two approaches in order to tackle this problem. In supervisory control theory, a *supervisor* takes on an active role in the execution of a system, and such a supervisor has control over certain events in the system, see [Ramadge and Wonham, 1987]. This approach, which will not be the topic of this thesis, can be useful whenever we have direct control over events in the environment, which is for example the case in a factory. A second approach, and the approach we consider in this thesis, is the use of regulative norms, which exhibit a more passive role within the system. Here, agents can decide whether to comply with the norm or not, and monitoring and sanctioning mechanisms exists in order to detect, and correct, the behaviour of the agents when needed. This approach is useful whenever there exist many events in the system which cannot be directly controlled. Normative (multi-agent) systems is an active research field within the field of Artificial Intelligence, and in the next chapter we will give a brief historical overview of the field.

In order to answer the second question, it is important to realise that these formal frameworks enable us to prove correctness of these systems, which is referred to as *formal verification*. Formal verification is the act of proving

(or disproving) that the system works as intended. The crucial notion here is *(dis)proving*, which is in contrast to other methods of testing a system. For example, in a simulation-based approach, a model of the system is constructed and simulated in the hope of finding, and eliminating, undesirable scenarios. However, with such an approach it is possible to miss or overlook bad scenarios, and thus we never have a guarantee that the system is correct. As these systems are growing in size and scale, it becomes easier to overlook these bad scenarios. A second important reason is that these formal frameworks allow us to perform this verification *offline*. This implies that we perform verification based on the specification of the system, meaning that we do not have to execute the system in order to prove its correctness. Whenever the cost of defection is high, it is of crucial importance that we know that a system is correct without actually having to run it. Offline formal verification gives us this guarantee. A third reason why formal verification is so important is because, as we will see, it leads to the development of logics and formal methods that allow us to reason about the properties of a system. Logics and formal methods are not only useful for the designer of a system, but also play a crucial role for the participating agents in order to verify whether a given system can be used to reach personal objectives and/or whether the participation in such a system is beneficial at all. Development of such methods and tools play an important role in the advancement of normative multi-agent systems and Artificial Intelligence in general.

Although formal analysis and verification has several desirable properties, it also has several drawbacks. Particularly, we want to focus the attention on the following:

- A formal framework usually focusses on a small relevant fragment of the entire system. That is, we typically omit facts about the system which we deem irrelevant to include. However, this over-simplification can also mean that the framework we consider is *too simple*, implying that we can miss relevant properties of the actual system.

- A formal framework can only be used if we can extract a formal specification of the system, the agents and the design objectives. In some cases this can be troublesome, as these specifications are often only partial, incomplete or unknown.

- Formal verification is (computationally) costly. For example, a typical model checking algorithm has to overcome the combinatorial blow up of the state-space, also known as the state explosion problem. A typical way to overcome this problem is for example to modularize the system and to perform verification on these separate modules. Although we do have several computational complexity results in this thesis, we are not

(yet) concerned with the specific algorithms to perform the verification tasks.

Moreover, it is important to note that formal verification is not an end to all means, and that typically verification of a system should be contrasted with other means of testing a system, for example a *validation* procedure.

1.2 Research questions

As previously mentioned, this thesis aims to design and analyse formal logic-based frameworks in which normative multi-agent systems can be modelled and verified. This section goes into more detail about the surrounding questions and difficulties related to this aim. In this thesis we study enforcement norms, which are norms which can be violated, and which can lead to sanctions when a violation is detected. Moreover, we consider multi-agent systems in which the specification and internal architecture of the participating agents are primarily unknown to us. This implies that we cannot simply assume that the agents are aware of the norms, or that they are compliant with respect to the norms. Moreover, agents may have preference over certain executions of the system, and these preferences may be unknown to us. However, once we make the assumption that the agents have some preference, and we are interested in how the agents behave, we enter the field of game theory. Game theory, in the broad sense, is the study of strategic decision making in the presence of (one or more) rational agent(s). Thus, a formal framework which allows us to model and verify normative multi-agent systems must be able to capture this intricate interplay between logic, norms and games that arises. This leads us to our first research question.

Research question 1. *How can we design and analyse logic-based frameworks which are able to capture the intricate interplay between logic, norms and games in order to model and verify normative multi-agent systems?*

As we will see, there is not one unique way of setting up such a framework, and this thesis will explore different approaches. Particularly, this question will be the focus of Chapter 3 and Chapter 4. These approaches consider that the normative system is a static entity, that is, that the set of norms contained in the system are static and fixed throughout the execution of the system. However, in a typical normative system these norms may change over time, and these approaches do not cope with this fact. In this thesis we also want to model and verify normative multi-agent systems situated in a dynamic environment. A formal framework to model and verify these systems must thus be able to characterize how these systems evolve over time.

Research question 2. *How can we design and analyse a logic-based framework to model and verify normative multi-agent systems situated in a dynamic environment?*

We will explore this question in Chapter 5. In order to control the behaviour of the system and the participating agents, it is not only important to consider the norms, but also the monitors that are able to detect whenever a violation occurs. In other words, both the norms and the monitors serve as a control mechanism for the multi-agent system, and this thesis will focus on both. Monitors are important whenever the system as a whole is not fully observable. A formal framework to analyse and verify these monitors must thus be able to characterize the interplay that arises between norms, monitors and the behaviour of the system.

Research question 3. *How can we design and analyse a logic-based framework to model and verify monitors observing the behaviour of a normative multi-agent system?*

This last question will be explored in Chapter 6. It is important to note that the frameworks we consider do not facilitate a complete list of all the possible verification questions a designer might ask. For example, one of the topics that we do not cover in this thesis is how sanctioning works, and how possible sanctioning mechanisms can be implemented. A research question related to this is how we can verify whether a sanctioning mechanism is correct with respect to the given norms in the multi-agent system. We do feel however that the above questions serve as a good starting point for the research of formal verification of normative multi-agent systems.

Another important problem related to verification is *synthesis* and *design*, which asks whether we can find a set of norms such that the design objectives of a system are met. Verification usually precedes synthesis, since in order to perform synthesis we first need to have a way of verifying when a synthesized solution is correct. This thesis does not cover design and synthesis questions, but serves as a stepping stone towards these questions.

1.3 Thesis outline

We give a brief outline of the contents of this thesis:

- In **Chapter 2** (p 15) we will explore the notions of multi-agent systems, norms and logic-based frameworks further by giving a brief background on these topics.
- In **Chapter 3** (p 27) we design and analyse a formal framework to model and verify normative multi-agent systems using compliance types. This chapter is related to Research question 1.

- In **Chapter 4** (p 53) we design and analyse a formal framework to model and verify normative multi-agent systems using principles from mechanism design. This chapter is related to Research question 1.
- In **Chapter 5** (p 109) we design and analyse a formal framework to allow us to model and verify the behaviour of a dynamic multi-agent system under addition of various classes of norms. This chapter is related to Research question 2.
- In **Chapter 6** (p 153) we design and analyse a formal framework to allow us to model and verify monitors situated in a normative multi-agent system. This chapter is related to Research question 3.
- In **Chapter 7** (p 183) we discuss this thesis.

1.4 Relation to earlier work

A lot of the content in this thesis is based on earlier work we have published. Although many of these ideas are either extended or further developed in this thesis, they can be traced back to the following publications:

- **Chapter 3** is based on [Knobbout and Dastani, 2012], in which we developed and presented our initial framework allowing us to reason about the strategic abilities of agents under various compliance types.
- **Chapter 4** contains novel work.
- **Chapter 5** is based on [Knobbout et al., 2014], in which we developed our initial framework allowing us to reason about dynamic normative multi-agent systems. The work in this thesis extends this earlier work in several noteworthy ways, particularly by providing an extended logical language and by providing an accompanying proof system.
- **Chapter 6** is based on [Bulling et al., 2013], in which we developed our initial framework in which monitors can be modelled and verified.

1.5 Not covered by this thesis

In this section we discuss some related topics which are not covered in this thesis. Note that some of the topics listed below are discussed in more detail in Chapter 2.

The norms we consider are enforcement norms, which are norms that are enforced by some (legal) authority and can possibly be violated by the agents, leading to a possible sanction. We do not consider other types of norms, for

example constitutive norms, which are norms that relate brute to normative aspects of a system, for example with the use of "counts-as" rules. Additionally, we do not consider complex norm structures like contrary-to-duty obligations. We also do not concern ourselves with social norms, which are standards of behaviour shared by members of a social group, and may include things like what honest behaviour in a particular situation is. Moreover, we are not concerned with how these norms may come into existence, which is for example studied in the field of norm emergence.

As mentioned previously, we are only concerned with verification of normative systems, and do not concern ourselves with synthesis and design of norms. We do not wish to focus our attention on how these norms can be implemented, that is, how enforcement and sanctioning strategies can be implemented, we merely assume that implementation of norms has certain effects on the system. For example, we assume that implementation implies that certain sanctions may come into effect, without worrying how this works on a procedural level. We do not consider other means of testing a system, for example validation procedures. Moreover, we do not concern ourselves with run-time verification, which considers the verification of running systems. Finally, we are not concerned with the specific algorithms to perform the verification tasks. However, we do have several computational complexity results.

In this thesis we take on a system perspective, and not an agent perspective. This means that we are not concerned with questions related to the internals of an agent, such as what the intentions, beliefs, desires and responsibilities of the agents are.

Chapter 2

Background

In this chapter we will explore the notions of multi-agent systems, norms and logic-based frameworks further by giving a brief background on these topics.

2.1 Multi-agent systems

A multi-agent system is a computerized system that is composed of multiple interacting agents within an environment [Wooldridge, 2009]. Such a system is generally composed or designed with a specific goal in mind. For example, an electronic marketplace consisting of several buyer and seller agents can be used to perform safe transactions between currency and goods. Agents are the actors within such an environment. Typically multi-agent research refers to software agents. However, the agents in a multi-agent system could equally well be robots or humans. Agents within a system may have diverging information, diverging interests, or both. Historically, a lot of research has been devoted in creating suitable frameworks in which the behaviour of such agents can be described, explained and predicted. Based on [Dennett, 1987] of using the intentional stance to describe the behaviour of an agent, [Bratman, 1987] developed the BDI (Beliefs, Desires, Intentions) theory of practical reasoning. In this theory, it is argued that an agent needs beliefs, desires and intentions in order to exhibit rational behaviour. Many attempts have been made to formalize these concepts in logic-based frameworks, noteworthy that of [Cohen and Levesque, 1990] and [Rao and Georgeff, 1991]. Consequently, this research has led to the development of agent-based programming languages, see for example [Shoham, 1993]. However, in this thesis we take on a system perspective, and are thus *not* interested in the internal architecture of an agent. This means that the knowledge, beliefs and intentions of the agents are unknown to us. Such systems can typically be modelled as *open systems*, which are systems that interacts with its environment and whose behaviour

depend on the state of the system as well as the behaviour of the environment [Alur et al., 2002]. An open multi-agent system is an open system whose environment consists of multiple agents.

2.1.1 Open multi-agent systems

An open system, as opposed to a reactive system, is a system that interacts with its environment and whose behaviour depend on the state of the system as well as the behaviour of the agents and the environment [Alur et al., 2002]. Often the definition of an "open system" is also used to denote systems in which there is no access to the internal architecture of the agent and in which it is not possible to predict the agents' interactions, see for example [Artikis and Sergot, 2010]. Observe that these definitions are not always interchangeable, since our definition of an open system does not imply that we cannot predict the agents' interactions. Moreover, sometimes "open systems" are referred to systems in which the agents may join and leave, which is an entirely different concept altogether. When specifying an open system, we need to somehow specify how the system interacts with its *environment*. The environment of an open multi-agent system consists of agents, so the specification must also incorporate what the agents can do (the possible actions of the agents), and how these possible actions affect the overall state of the system. Depending on the kind of open systems we consider, the environment can have several characteristics, notably (but not exclusively) [Wooldridge, 2009]:

- **Concurrent:** The environment of a system is *concurrent* whenever agents have the ability to act concurrently, i.e. the agents can perform actions simultaneously. Another possibility is that the agents act *interleaved* or *turn-based*, i.e. the environment forces the agents to act in turns.

- **Synchronous:** The environment of a system is *synchronous* whenever the agents act synchronously. In synchronous systems, a centralized scheduler decides when the agents get to perform their action.

- **Decentralized:** The environment of a system is *decentralized* whenever there is no central controller that forces the agents to coordinate their actions. Another way of framing this is that agents may act completely independent of each-other.

- **Discrete:** The environment of a system is *discrete* when the characteristics of the system can be represented by a discrete sets. In these kinds of systems, facts can typically be represented by a set of atomic propositions, and actions can be enumerated by some alphabet.

- **Deterministic:** The environment of a system is *deterministic* when the actions of the agents completely and uniquely decide how the system progresses between states.

In this thesis, in order to represent an open multi-agent system, we use (labelled) *transition systems* to compactly represent the semantics of a system, which can be seen as a unifying approach among different ways of specifying a multi-agent system, see for example [Alur et al., 2002]. A transition system in its most simplistic form can be thought of as a graph, where the vertices represent the possible states of the system, and the edges represent the possible actions of the agents. A traversal of such a graph thus represents a possible computation of the system. Even though such a representation is fairly simplistic, it is still able to capture various characteristics of an open system we mentioned above. For example, if we want to model a concurrent system, we can label the edges of a transition system with tuples of actions, and if we want to model non-determinism, the same edge-label can lead to several different states. As we will see, in each chapter we will use slightly different instantiations of a transition system depending on the features of the model we are interested in. In the beginning of each chapter we will make it explicit what kind of transition system we consider, and what the various characteristics of the system are. To give a brief overview, in Chapter 3 we use concurrent game structures to model concurrent systems, in Chapter 4 we use turn-based structures derived from a set of actions which are given in terms of pre- and post- conditions (e.g. a STRIPS program, see [Fikes and Nilsson, 1971; Russell and Norvig, 2003]), in Chapter 5 we use labelled transition systems where the labels are used to identify the (atomic or concurrent) actions, and in Chapter 6 we use non-labelled transition systems. Again we stress that these are all instantiations of a transition system, and that different models were chosen to model different features of a system.

2.2 Norms

Norms, in its most general form, are standards of behaviour, and include things such as obligations, permissions and prohibitions. A typical norm in a smart road multi-agent system might for example be that the agents ought to stop whenever there is a red sign. Norms have been proposed in multi-agent systems to deal with coordination and security issues that can arise, see for example [Boella et al., 2006]. Historically, norms have been thoroughly studied by legal theorists and philosophers in the context of *normative systems* and *deontic logic*. A normative system is a system that represents the things that *should* (or *should not*) be the case, and which requires normative concepts to be described or specified [Alchourrón and Bulygin, 1971; Meyer and Wieringa,

1994]. Given such a system, deontic logic can be used to reason about these normative concepts, i.e. the permissions, obligations and prohibitions that follow from such a system. Deontic logic has its origins in the work of [von Wright, 1951], which formally studied the concept of "oughts", leading to the system of Standard Deontic Logic (SDL). Deontic logic has since then evolved from this simple logical system to more complex logical systems, and found its use in various areas, notably in computer science [Meyer and Wieringa, 1994]. Various authors around this time played with the idea of using (simple) normative systems together with a multi-agent system in order to create, what we now refer to as, *normative multi-agent systems*. Normative multi-agent systems are multi-agent systems governed by a normative system, and before we move on to this topic, it is worth mentioning that there is a whole body of research on norms from a sociological perspective, see for example the work of [Therborn, 2002] for an overview. This thesis studies norms from a computer science perspective, in which we use the presumption that norms can be explicitly represented by some logical or formal construct.

2.2.1 Normative multi-agent systems

A normative multi-agent system is a multi-agent system together with a normative system. As previously mentioned, norms have been proposed in multi-agent systems to deal with coordination and security issues that can arise. In general, norms in a multi-agent system can serve different functional purposes. For one, norms can be distributive of nature, i.e. they define how costs, risks and rewards are allocated among society. Secondly, norms can be constitutive of nature, i.e. they relate brute to institutional aspects of the system for example with the use of "counts-as" rules, see [Searle, 1969]. Lastly, norms can be regulative of nature, that is, they define what the agents or the system must and must not do. This work is only concerned with this last kind, and as such the norms in this thesis all either refer to some behaviour of the agents, or refer to some behaviour of the system. Regulative norms can again be subdivided into many categories, such as legal, social and moral norms, many of which have significant overlap with each other. In this thesis we do not want to make any strong claims regarding these categorizations, we merely assume that there is an enforcing entity involved, often referred to as *enforcement* norms. In [Grossi et al., 2007], the following distinction is made between enforcement norms on the one hand, and regimentation norms on the other:

- **Enforcement norms** are norms which are enforced by some enforcing mechanism. An example of an enforcement norm is the speed limit on a smart road system, which is enforced by the legal authority which sanctions an agent if a violation is detected.

- **Regimentation norms** are norms which are regimented by constraints in the environment. An example of a regimentation norm is the paying of a parking ticket, which is regimented by a barrier blocking the exit of the parking lot.

Thus, the important difference between the two is that enforcement norms can be *violated*, while regimentation norms can not. In this thesis, we will exclusively focus our attention on these enforcement norms. Typically with such norms, whenever the system as a whole is not fully observable, we require monitors in order to detect whenever a violation occurs. Thus, both the norms and the monitors serve as a control mechanism for the multi-agent system, and in this thesis we will not only focus on norms, i.e. Chapter 3, 4, 5, but also on monitors and its relation to norms, i.e. Chapter 6. Similarly as in the work of [Anderson, 1958] and [Meyer, 1987], we can additionally associate with an enforcement norm a propositional atom which would be the case if a violation occurs. Such an atom can be used to signal that something bad has happened. In the general case, such a violation can also be the instantiation of another norm (for example *contrary-to-duty obligations*, see [Prakken and Sergot, 1996]), which will not be covered by this thesis. Moreover, we also do not cover how enforcement can be implemented, and thus we often make the simplification that a violation is immediately followed by a sanction issued by some enforcement mechanism. These propositional sanctions will play an important role in Chapter 4, where they can influence the behaviour of the agents by assuming that the agents have preferences over these sanctions. Moreover, they will play a role in Chapter 5, where we will develop a logical language of norm change allowing us to reason about the sanctions that come into effect after certain executions of the system.

As mentioned previously, the norms we consider are regulative. However, in the kind of systems we are interested in the specification and internal architecture of the participating agents might be unknown to us. This implies that we cannot simply assume that the agents are aware of the norms, or that they are compliant with respect to the norms. Thus, similarly as in the work of e.g. [Sergot, 2007], [van der Hoek et al., 2007], [Dastani et al., 2011] or [Tinnemeier et al., 2009], we treat norms as some artefact of the environment, and do not study norms in relationship to the internals of an agent, which is for example the topic of the work in [Verhagen, 2000]. As stated before, we can additionally associate with an enforcement norm a propositional sanction which would be the case if a violation occurs. By our assumption that there exists some mechanism which enforces these sanctions whenever a violation occurs, we can also view the addition of a norm as an *update* of the specification of a multi-agent system. In this updated system, the states of the system not only consist of the propositional facts of the system, but also of

the various propositional sanctions that occur whenever there is a violation. We can use such an updated system to prove several properties of a normative multi-agent system, which is the approach we will take in Chapter 4 and Chapter 5. In contrast to this, in Chapter 3 and Chapter 6 we treat norms as a separate semantic artefact pertaining to the permitted and forbidden behaviour of the multi-agent system. We however stress that even though we use different perspectives in order to analyse and reason about the effects of norms on multi-agent systems, we are still within the scope of open systems and the effects of enforcement norms on these systems.

The norms we consider are regulative, that is, they define what the agents or the system must and must not do. As such, the norms in this thesis all either refer to some behaviour of the agents, or refer to some behaviour of the system. When examining related research, several *classes* of these norms can be identified, ranging from simple states to complex action sequences that should be achieved or avoided. In this thesis, we identify the following classification of regulative norms:

- **State-based norms** (see [Sergot, 2007; Grossi, 2007; Dastani et al., 2009a,b]): A state-based norm refers to certain states that should be achieved or avoided. These kinds of norms are of the 'to-be' variant, i.e. they refer to certain facts that should, or should not, be the case. These norms are for example studied in the work of [Sergot, 2007], leading to a certain 'colouring' of the states of the system (green and red states). We analyse these norms in Chapter 4, where we view state-based norms as a special kind of sequence-based norms (see 'sequence-based norms' below), and Chapter 5, in which we develop logical languages in order to express and reason about this class of norms.

- **Action/transition-based norms** (see [Meyer, 1987; Shoham and Tennenholtz, 1992; Woolridge and van der Hoek, 2005]): An action-based norm refers to actions that should be achieved or avoided, while a transition-based norm refers to transitions that should be achieved or avoided. Although the difference is minor, a transition can be caused by a multitude of actions (two different actions might result in the same state), while an action can result in a multitude of transitions (because of concurrency or non-determinism). This kind of norms are of the 'to-do' variant, i.e. they refer to certain actions or transitions that should, or should not, be performed. These norms are for example studied in the work of [Shoham and Tennenholtz, 1992] and [Woolridge and van der Hoek, 2005], where norms are viewed as a subset of the possible actions or transitions that can occur. We analyse these norms in Chapter 3, and in Chapter 5 we develop logical languages to express and reason about this class of norms.

- **Sequence/process-based norms** (see [Meyer, 1987; Alechina et al., 2013; Broersen, 2004]): A sequence- or process-based norm refers to certain sequences of states or actions that should be achieved or avoided, and are for example discussed in [Meyer, 1987]. Sequence-based norms can be seen as a generalization of state- and transition-based norms, since for example the state-based norm forbidding a single state can be encoded by a sequence-based norm in which every possible sequence containing this state is forbidden. These norms are analysed in Chapter 4.

- **Computation/run-based norms** (see [Bulling et al., 2013; Alechina et al., 2015]): A computation- or run-based norm refers to certain infinite sequence of actions or states that should be achieved or avoided. They can be a useful class to study, since they generalize the concept of state-based, transition-based and sequence-based norms even further, and are for example studied in the work of [Alechina et al., 2015]. These norms are analysed in Chapter 6, where we specify these norms in terms of Linear-time Temporal Logic formulas.

As remarked in [Boella et al., 2006], there is as of now still no agreement among the researchers in the normative multi-agents systems community on how to represent such norms. For this reason we take great liberty in each chapter on how we represent the norms, particularly in Chapter 3 we treat them as a subset of the possible transitions that can occur in the system, in Chapter 4 we treat them as functions from histories to sanctions, while in Chapter 5 and Chapter 6 we treat them as sentences of some logical language. Although these representations differ, it is important to remember that we are still referring to the similar category of enforcement norms. In the beginning of each chapter we will make it explicit how the norms are represented.

2.3 Logic-based frameworks

In this thesis, we use logic-based frameworks in order to model and verify normative multi-agent systems. This verification process is often referred to as *logic-based formal verification*. Formal verification is the act of rigorously proving (or disproving) that a system works as intended, that is, with respect to a formal specification or property, see [Schneider, 2004] and [Huth and Ryan, 2004] for an introductory overview. Such a property is understood as the intended goal of a designer, but might equally well be some social welfare criterion, representing what is good/optimal for the entirety of society (i.e. the entire collection of participating agents). Typically such properties are either temporal claims of *liveness* or *safety* of a system. A property of liveness expresses that eventually something good will happen (e.g. eventually the

goals of all the agents are reached), and a property of safety expresses that nothing bad will happen (e.g. the system will not crash). In this thesis, as opposed to run-time verification, we perform this verification *offline*. In offline verification, we are given (1) the specification of the open multi-agent system, (2) the specification of the enforcement norms and (3) the specification of a desired property, and are asked to verify whether the system and the norm adhere to the desired property.

As previously mentioned, the norms we are interested in can be violated by the agents. However, we cannot simply assume that the agents are aware of, or compliant with respect to, the norms. In other words, an important question that needs to be answered is how we can perform formal verification if we have no idea whether or not the agents will be norm-compliant. This thesis seeks to answer this question, and the general strategy we adopt is as follows:

1. Firstly, we abstract from the specification of the normative multi-agent system the possible behaviours which can occur. A behaviour in this thesis is understood as either a finite history or an infinite computation of the system. Such a behaviour can be referred to as a possible *outcome* of the system.

2. Secondly, we look at some relevant behaviours which we predict will happen when considering the norms. In other words, the norms act as a *refinement* of the set of possible outcomes. Of course, there is not one unique way of applying such a refinement, and in this thesis we consider the following general approaches:

    - In Chapter 3 we consider compliance types. Given a set of behaviours and given such a compliance type, we only consider the behaviours which are aligned with this type. Such a compliance type may for example state that all the agents are obedient, or that some of the agents are disobedient while the remaining are. For example, when considering a smart road multi-agent system, it might be meaningless to consider that all the agents are disobedient (e.g. every agent ignores every red light), but it might be meaningful to consider that *some* of the agents might be doing that. In this approach, the designer can choose which compliance type might be relevant to consider.
    - In Chapter 4 we consider agent types in combination with game theoretic solution concepts. An *agent type* tells us what an agent knows and what an agent values, and since we might not know the true type of an agent, multiple types can be considered for a single agent. A possible assignment of these types to agents gives rise to

a possible *game* the agents might be playing, which means we enter the field of game theory [Neumann and Morgenstern, 1944]. In this approach, a norm can, by adding sanctions to a possible behaviour, make certain behaviours more attractive while others less attractive. A *solution concept* is a formal rule predicting how the agents will behave in such a game, most famously the concept of a Nash equilibrium. Using such a solution concept in combination with the norm and agent types, we can extract the relevant behaviours which we predict will occur. In this approach a norm acts as a mechanism, and this approach is very closely related to the topic of *mechanism design*, see for example [Osborne and Rubinstein, 1994].

3. Finally, we verify whether this behaviour is aligned with the intended specification or property.

In order to reason about the correctness of a normative multi-agent system, we develop in Chapter 3 and Chapter 4 a logical language in which we can specify these verification claims. Thus, the crucial difference in our approach as opposed to "standard" verification lies in the second step where we apply our refinement. This kind of verification is considering the entirety of the normative multi-agent system, and seeks to answer questions related to the system as a whole. In contrast to this kind of verification, we are also concerned with modelling and verifying dynamic normative systems in Chapter 5, and with modelling and verifying monitors observing the behaviour of normative multi-agent system in Chapter 6:

- In Chapter 5 we ask the question, given a normative multi-agent systems situated in a dynamic environment, whether a specific behaviour has the desired normative effects. Such a behaviour in this setting can consist of actions and normative updates that can take place. This allows us to verify whether the dynamic normative multi-agent system is correct with respect to the behaviour we are interested in.

- In Chapter 6 we ask the question whether, given a behaviour of the system and a formal specification of a *monitor*, it is the case that when a violation is detected the violation actually occurred, and when a violation is not detected, the violation did not occur. Moreover, we ask whether this is the case for the set of all possible behaviours of the system, i.e., whether a monitor is both *sound* and *complete* for a system. This allows us to verify whether a monitor is sufficient for a normative system, i.e. whether it soundly and completely detects the possible norm violations that can occur.

Again, in order to reason about the correctness of a dynamic normative multi-agent system, or a monitor observing the behaviour of a multi-agent system,

we develop in Chapter 5 and Chapter 6 a logical language in which we can specify exactly the various verification claims we would like to be answered. As such, formal verification of a normative system reduces to verifying whether a specific formula from this logical language is *satisfied* in a specific system. In order to perform this task we can use *model checking* and *deductive verification*.

2.3.1 Model checking

Model checking refers to the approach and associated problems of exhaustively and automatically checking whether a given model meets a given specification, for an overview see [Clarke et al., 2001] and [Baier and Katoen, 2008]. In this work, the model is a transition system representing the open multi-agent system, and the specification is represented by a logical assertion. Model checking is a computational problem, and typically has the following questions associated to it:

1. What is the complexity class associated with the decision problem? Answering this question allows us to determine the inherent difficulty of the problem, and shows us how this problem relates to other problems by placing it in the computational hierarchy. Such a classification allows us to analyse the complexity of a task regardless of the specific algorithm that is being used.

2. What is an efficient algorithm to solve the problem? For example, there exists multiple way of enumerating the reachable states of a model. In *explicit-state model checking*, the states are enumerated one at a time, while in *symbolic model checking*, the state space traversal is based on a symbolic representation of the states.

In this thesis we focus on the first question, which allows us to show how model checking problems of normative multi-agent systems relate to similar more generic model checking problems. We apply these techniques in Chapter 3 and Chapter 6.

2.3.2 Deductive verification

Deductive verification refers to the general approach and associated problems of constructing a mathematical/logical *proof system* in which axioms and rules can be used in order to construct a proof for the validity of the desired property within the model. An introduction to the principles of this technique can be found in [Mendelson, 1987]. Generally speaking, this approach is characterized by the following steps:

1. First, we construct a calculus consisting of axioms and rules. An axiom is a premise or starting point of reasoning, leading to a collection of assertions which we are evidently true. Rules can be used to infer from old assertions new assertions. An assertion is provable within the calculus if, by applying the axioms and rules of the system, we can infer the assertion.

2. Second, we analyse whether such a calculus is suitable for our logic. An important criterion could be that a proof system is *consistent*, i.e. it should not be able to prove that an assertion is both true and false. Two other important criteria are that of *soundness* and *completeness*. A proof system is sound if and only if everything that is provable is true, and a proof system is complete if and only if everything that is true is provable. There are several other criteria that are important, for example that of *decidability*, which refers to the question of whether there exist an effective method for determining whether an assertion is provable. Later in this thesis we will delve deeper into these criteria.

We construct proof systems in Chapter 4 and Chapter 5, for which we show soundness, completeness and decidability.

Chapter 3

Modelling and Verifying using Compliance Types

In this thesis, we consider open multi-agent systems, which are systems in which the specification and internal architecture of the participating agents are unknown to us. This implies that we cannot simply assume that the agents are aware of the norms, or that they are compliant with respect to the norms. In other words, a crucial question that needs to be answered is how we can perform formal verification if we have no idea whether or not the agents will be norm-compliant. In order to tackle this problem, we consider compliance types in this chapter. Such a compliance type may for example state that all the agents are obedient, or that some of the agents are disobedient while the remaining are obedient. This will be the topic of this chapter, where we will explore how we can verify normative multi-agent systems using these compliance types.

3.1 Introduction

The approach we consider in this chapter is related to the approach found in [Shoham and Tennenholtz, 1992] and [Moses and Tennenholtz, 1995], where they study the concept of social laws. In these works, social laws are used to constrain the behaviour of the agents by forbidding certain actions in specific situations. An important distinction with social laws and the kind of enforcement norms we consider in this thesis are that social laws are *regimented*. In other words, social laws cannot be violated by the agents, while enforcement norms can. This line of research was later extended by several authors in a multitude of ways. For example, in [van der Hoek et al., 2007] the basic idea is extended to the more expressive modelling domain of Alternating-time Temporal Logic (ATL) with the idea that the addition of a social law to a

multi-agent system is successful if, by implementing it, the overall objective of the system is satisfied. Later these ideas were extended with the notion of preference to reason about norm compliance in normative systems. Examples of such extensions are [Woolridge and van der Hoek, 2005], where each agent was attributed a single goal, and [Ågotnes et al., 2007], where agents were given a priority list of goals. The main topic of these papers was to investigate whether certain sets of norms can be considered 'stable' under consideration of the agents' preferences.

In our framework, we consider *transition-based norms*, which we already briefly introduced in Chapter 2. These kind of norms are of the 'to-do' variant, i.e. they refer to certain transitions that should, or should not, be performed by certain (coalitions of) agents. A typical transition-based norm might state that "coalition A ought to avoid going from state q_{from} to state q_{to}". These norms are similar to social laws in the sense that they denote the disallowed transitions of the system. However, these norms are different from social laws in the sense that the agents are allowed to violate them. Moreover, and in contrast to social laws, norms are directed to coalitions of agents and not only to individual agents. Thus, we do not assume that "implementing" a system of norms enforces every agent to be perfectly norm obedient. To some extent, our research is related to [Ågotnes et al., 2010] in which the need for robust normative systems is discussed. They introduce an extension of Computation Tree Logic (CTL) in which statements such as "if coalition A is norm compliant, then this is sufficient to guarantee φ" can be expressed. In their work a robust normative system is defined as a multi-agent system which remains 'effective' even if certain agents behave in a non-compliant manner.

The approach we take in this chapter is to introduce an extension of ATL to reason about properties of normative multi-agent systems under various compliance types. A compliance type may for example state that "the entirety of a coalition and all its sub-coalitions are always obedient with respect to the norms", or "there exists an agent in this coalition which is always disobedient with respect to the norms". Our proposed extension of ATL can be used to specify and verify whether the overall objective of multi-agent systems can be satisfied under the assumption that a coalition of agents behave according to a specific compliance type while the agents outside the coalition behave according to another compliance type. This approach is related to the approaches found in [Bulling et al., 2009; Jamroga et al., 2005; van der Hoek et al., 2005], in which standard ATL is enriched with more sophisticated strategic quantifications taking into account the intentions, commitments or rationality assumptions of the agents. Our extension can be used by the designers of a normative multi-agent systems to verify whether the overall objectives of the system are satisfied under the assumption of various compliance types. This extension can also be used by individual agents who need to reason and decide

if they can achieve their own objectives by behaving according to a specific compliance type in a normative system when the system is populated by other agents that behave according to another compliance type. This chapter is an extended work of our earlier published work found in [Knobbout and Dastani, 2012].

3.1.1 The verification problem

The verification problem we consider in this chapter is the following: Given the specification of a multi-agent system, a set of transition-based norms, and the design objectives pertaining to the liveness or safety of the system, verify using strategic reasoning with various compliance types whether the design objectives are satisfied in the system. The corresponding synthesis and design question is whether we can synthesize a norm set such that these design objectives are met.

3.1.2 Chapter outline

We give a brief outline of the contents of this chapter:

- In **Section 3.2** (p 29) we introduce the model of execution we are interested in, give a brief introduction to Alternating-time Temporal Logic (ATL) and define the notion of transition-based norms.
- In **Section 3.3** (p 35) we introduce our compliance types and develop our extension of ATL that allows us to reason about the strategic abilities of (coalitions of) agents under various compliance types. We show several properties and validities of this logic, and end this section with an elaborate example.
- In **Section 3.4** (p 44) we explore the model checking questions related to this logic. Particularly, we explore the computational complexity of verifying whether a formula is valid in a given system.
- In **Section 3.5** (p 48) we show how we can use this logic to prove several aspects of a multi-agent system. Particularly, we explore how we can prove properties of norm sets using this logic.
- In **Section 3.6** (p 51) we discuss this chapter.

3.2 Preliminaries

This section introduces the model of execution we are interested in. Moreover, we introduce the logic of Alternating-time Temporal Logic (ATL) to reason

about the strategic abilities of agents in a given concurrent game structure. We end this section by introducing the notion of transition-based norms which we will use throughout this chapter.

3.2.1 Concurrent game structures

A concurrent game structure, or CGS, can be seen as a multi-agent extension of a simple transition system as discussed in [Alur et al., 2002]. We will use these game structures throughout this chapter as the underlying model for an open multi-agent system. It consists of states of the world and a complete labelling of joint-actions over the transitions connecting these states. Formally, a CGS is defined as follows.

Definition 3.1 (Concurrent game structure (CGS))**.** *A concurrent games structure S is a tuple $(Q, \Pi, \mu, Ags, act, \delta)$ such that:*

- *Q is a finite set of states.*
- *Π is a finite set of atomic propositions.*
- *$\mu : Q \to \mathcal{P}(\Pi)$ is a valuation function mapping a state $q \in Q$ to an element from $\mathcal{P}(\Pi)$.*
- *$Ags = \{1, ..., n\}$ is a finite non-empty set of agents.*
- *$act : Ags \times Q \to \mathbb{N}_{>0}$ is a function that assigns to each agent and each state the number of available actions. We identify the actions of agent $i \in Ags$ in state $q \in Q$ with the numbers $1, \ldots, act_i(q)$. For each state $q \in Q$, a joint action is a vector $\vec{\alpha} = (\alpha_1, \ldots, \alpha_{|Ags|})$ such that $1 \leq \alpha_i \leq act_i(q)$ for every agent $i \in Ags$. Given a state $q \in Q$, we write $Act(q)$ for the set $\{1, \ldots, act_1(q)\} \times \cdots \times \{1, \ldots, act_{|Ags|}(q)\}$ of all possible joint actions.*
- *δ is a transition function which maps a state $q \in Q$ and joint action $\vec{\alpha} = (\alpha_1, \ldots, \alpha_{|Ags|}) \in Act(q)$ to the resulting next state $\delta(q, \vec{\alpha}) \in Q$.*

Relating to the various properties of multi-agent systems we discussed in Chapter 2, this model is thus concurrent, synchronous, decentralized, discrete and deterministic. This implies that at any moment in time each agent needs to decide on an action synchronously. Moreover, the performance of a joint action completely and uniquely defines the next state. Observe that every agent always has an applicable action available to him. Whenever an agent in a given state has no influence over the possible next state that can occur, we thus assume that this agent only has one applicable action in this state. Note that we have adopted a slightly different definition of a CGS when compared to [Alur et al., 2002, p677]. Particularly, they refer to *players* and *moves*,

while we refer to *agents* and *actions*. A CGS gives rise to a set of possible behaviours, which we refer to as the *runs* of the system.

Definition 3.2 (Runs)**.** *Given a concurrent game structure* $S = (Q, \Pi, \mu, Ags, act, \delta)$*, a* run *is defined as an infinite sequence of states* $q_0 q_1 q_2 \cdots \in Q^\omega$ *such that* $\forall n \geq 0$ *there exists a joint action* $\vec{\alpha} \in Act(q_n)$ *such that* $\delta(q_n, \vec{\alpha}) = q_{n+1}$*. For a given run* r*, we define* $r[n]\,(n \geq 0)$ *as the* n*-th state* q_n *occurring on the run, and define* $r[0, n]\,(n \geq 0)$ *as the finite sequence of states* $q_0 \dots q_n$*. The set of all possible runs over* S *is denoted by* $\mathcal{R}_S$*.*

Agents in the game structure can adopt *strategies* in order to enforce certain runs to occur. Informally, a strategy for an agent is a mapping from a finite sequence of states to an action. Whenever an agent or a coalition of agents adopts a certain strategy, they can enforce a set of runs to occur. This set of runs are all the runs that can happen independent of what the remaining agents do in the game structure. We often call such a set an *outcome* set. Formally, strategies and outcomes are defined as follows.

Definition 3.3 (Strategies, Outcomes)**.** *Given a concurrent game structure* $S = (Q, \Pi, \mu, Ags, act, \delta)$*, a* strategy *for an agent* $i \in Ags$ *is a mapping* σ_i*, mapping a finite sequence of states* $q_0 \dots q_k \in Q^+$ *to an element of* $\{1, \dots, act_i(q_k)\}$*. Given a set of agents* $A \subseteq Ags$ *and a state* $q \in Q$*, let* $\Sigma_A = \{\sigma_i \mid i \in A\}$ *be a set of strategies for* A*. A run* $r \in \mathcal{R}_S$ *is in* $out_S(q_0, \Sigma_A)$ *if it holds that for all positions* $n \geq 0$ *there is a joint action* $(\alpha_1, \dots, \alpha_{|Ags|}) \in Act(r[n])$ *such that for all* $i \in A$ *it is the case that* $\sigma_i(r[0, n]) = \alpha_i$*.*

Thus, we define $out_S(q, \Sigma_A)$ to be the set of runs starting from state q which the agents in A can enforce by following their respective strategies independent of what the agents $Ags \backslash A$ do. This set allows us to reason about the strategic abilities of agents, particularly it allows us to reason about what the agents can, and cannot, *enforce*. Alternating-time Temporal Logic was introduced to reason about these strategic abilities.

3.2.2 Alternating-time Temporal Logic

Alternating-time Temporal Logic, as introduced in [Alur et al., 2002], is a language to reason about the possible *runs* that agents can *enforce*. The language is interpreted over concurrent game structures, and allows assertions of the form $\langle\!\langle A \rangle\!\rangle \psi$, where $A \subseteq Ags$ is a coalition of agents and ψ is a temporal formula. Such a formula can be read as "coalition A can enforce ψ to be true". To evaluate such a formula at a state, we can consider a game between a protagonist and an antagonist which results in a run. At every round the protagonist chooses for each agent in A a move, and then the antagonist proceeds by choosing for every agent $Ags \backslash A$ a move, after which a transition occurs to

a next state. This process is repeated indefinitely, which results in a run r. The protagonist wins the game if the resulting run satisfies the sub-formula ψ, otherwise the antagonist wins. The formula $\langle\langle A\rangle\rangle\psi$ is satisfied at a state if the protagonist has a winning strategy for this game. Depending on the syntax used for ψ, we obtain different variants of Alternating-time Temporal Logic. If ψ is an arbitrary formula from Linear-time Temporal Logic (LTL) as proposed in [Pnueli, 1977], we obtain the language ATL*. If ψ consists of a single temporal operator applied to a state predicate, we obtain the language ATL. By allowing nesting of alternating properties, we obtain ATL as the alternating-time generalization of CTL, and ATL* as the alternating-time generalization of CTL*. Computation Tree Logic (CTL), as shown in [Clarke and Emerson, 1982], is a logic that allows explicit (universal and existential) quantification over the set of runs. To see why ATL generalizes CTL, observe that $\langle\langle\varnothing\rangle\rangle$ translates to a a universal path quantifier (since coalition $\varnothing$ cannot enforce anything) and $\langle\langle Ags\rangle\rangle$ translates to an existential path quantifier, since Ags can enforce every run.

In this chapter we focus on the basic language of ATL, since as shown in [Alur et al., 2002] ATL possesses several computation benefits over ATL*. Formally, given a concurrent game structure $S = (Q, \Pi, \mu, Ags, act, \delta)$, ATL is characterized by the following grammar, where $p \in \Pi$ and $A \subseteq Agt$:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\langle A\rangle\rangle\bigcirc\varphi \mid \langle\langle A\rangle\rangle\,\square\,\varphi \mid \langle\langle A\rangle\rangle\varphi\mathcal{U}\varphi$$

A formula of this language is evaluated along a concurrent game structure and a state in the following way.

Definition 3.4 (ATL semantics)**.** *Given a concurrent game structure $S = (Q, \Pi, \mu, Ags, act, \delta)$ and a state $q \in Q$, the satisfaction relation $\vDash_{ATL}$ of ATL is inductively defined as follows:*

- $S, q \vDash_{ATL} p$ *iff* $p \in \mu(q)$.
- $S, q \vDash_{ATL} \neg\varphi$ *iff* $S, q \nvDash_{ATL} \varphi$.
- $S, q \vDash_{ATL} \varphi_1 \vee \varphi_2$ *iff* $S, q \vDash_{ATL} \varphi_1$ *or* $S, q \vDash_{ATL} \varphi_2$.
- $S, q \vDash_{ATL} \langle\langle A\rangle\rangle\bigcirc\varphi$ *iff there exists a strategy set* Σ_A *for* A *such that for every run* $r \in out_S(q, \Sigma_A)$ *it holds that* $S, r[1] \vDash_{ATL} \varphi$.
- $S, q \vDash_{ATL} \langle\langle A\rangle\rangle\,\square\,\varphi$ *iff there exists a strategy set* Σ_A *for* A *such that for every run* $r \in out_S(q, \Sigma_A)$ *and all positions* $i \geq 0$ *it holds that* $S, r[i] \vDash_{ATL} \varphi$.
- $S, q \vDash_{ATL} \langle\langle A\rangle\rangle\varphi_1\mathcal{U}\varphi_2$ *iff there exists a strategy set* Σ_A *for* A *such that for every run* $r \in out_S(q, \Sigma_A)$ *there exists a position* $i \geq 0$ *such that for all positions* $0 \leq j < i$ *it holds that* $S, r[j] \vDash_{ATL} \varphi_1$ *and* $S, r[i] \vDash_{ATL} \varphi_2$.

A formula of the form $\langle\!\langle A \rangle\!\rangle \psi$ should intuitively be read as "Coalition A has a strategy in order to enforce ψ", where ψ can be a temporal formula of the form $\bigcirc\varphi$, to be read as "in the next state φ", $\Box\varphi$, to be read as "always in the future φ" and $\varphi_1 \mathcal{U} \varphi_2$, to be read as "$\varphi_1$ until φ_2 starts to hold". For example, the assertion $S, q \vDash_{\text{ATL}} \langle\!\langle A \rangle\!\rangle \Box p$ states that in concurrent game structure S and at current state q, it is the case that coalition A has the strategic ability to enforce p to always be true. In the general case, the ability for A to enforce something implies that $Ags \backslash A$ can do nothing to avoid this.

3.2.3 Transition-based norms

In this work we consider transition-based norms. Transition-based norms, in the most general sense, refer to certain transitions in the system that should be taken or avoided. The use of transition based norms is inspired by a multitude of previous work, in particular the "social laws" paradigm as presented in [Shoham and Tennenholtz, 1992] and [Moses and Tennenholtz, 1995]. As mentioned in the introduction, this work is extended in many directions, such as the work presented in [Woolridge and van der Hoek, 2005] and in [Ågotnes et al., 2007]. However, our work in this chapter differs from these approaches on the following key aspects:

1. Our norms, when introduced to a system, can still be violated. They are *enforcement norms* and are thus *not* hard-constraints on the behaviour of the agents.

2. Our norms can be directed towards a non-empty coalition of agents, and not just a single agent.

3. Our norms are transition-based and *not* action-based. We will elaborate on this later.

Formally, transition-based norms are defined in our framework as follows, where we write $\mathcal{P}_{\geq 1}(\dots)$ for the set of all non-empty subsets of a given set.

Definition 3.5 (Transition-based norms)**.** *Given a concurrent game structure* $S = (Q, \Pi, \mu, Ags, act, \delta)$*, a* transition-based norm *is a tuple* $\gamma = (q_{from}, A, q_{to}) \in (Q \times \mathcal{P}_{\geq 1}(Ags) \times Q)$*, where* $\mathcal{P}_{\geq 1}(Ags)$ *is the set of non-empty subsets of* Ags*. Given a set of transition-based norms* Γ *and non-empty set of agents* $A \subseteq Ags$*, we let* $\Gamma_A = \{(q_{from}, A', q_{to}) \in \Gamma \mid A' = A\}$ *be equal to* Γ *restricted to* A*.*

Throughout this chapter, we will refer to a set Γ consisting of transition-based norms as simply a *norm set*. The reading of a tuple $(q_{\text{from}}, A, q_{\text{to}})$ should be: "agents A ought to avoid going from q_{from} to q_{to}". Note that these norms

only relate to transitions that are forbidden. Given the above definition, we can identify along a run when a violation occurs.

Definition 3.6 (Enabled, taken, violated)**.** *Given a concurrent game structure S, a run $r \in \mathcal{R}_S$ and a transition-based norm $\gamma = (q_{from}, A, q_{to})$, we say that γ for a position $i \geq 0$ is ...*

- ...enabled *iff $i \neq 0$ and $r[i-1] = q_{from}$.*
- ...taken *iff $i \neq 0$ and $r[i] = q_{to}$.*
- ...violated *iff γ is enabled at i and γ is taken at i.*

Although transition-based norms, which refer to certain transitions that should be taken of avoided, and action-based norms, which refer to certain actions that should be performed or neglected, may appear similar, there is a subtle difference between the two. In a game structure, an action of an agent may lead to a multitude of possible next states, depending on what the remaining agents perform. Thus, if we would have an action-based norm stating that performing such an action is forbidden, all the possible next states of this action should result in a violation, which may, or may not, be desired. However, with transition-based norms we *can* make a distinction between the possible next states by simply stating which of these state-transitions are forbidden. On the other hand, certain state-transitions can be caused by a multitude of different joint actions. By using state-based norms, we cannot differentiate between these joint actions, while action-based norms *can*. In other words, transition-based and action-based norms can express different kinds of desired/undesired behaviours of a system. Since we are using concurrent game structures to model the behaviour of a multi-agent system, there is also a technical reason to use transition-based norm as opposed to action-based norms. The reason is that in this framework we want to assert whether a run (consisting of an infinite sequence of states) contains violations. However, if such a run does not keep track of the performed actions, we cannot assert whether such a run contains a violation. Note that the "abstract normative constraints" presented in our earlier work in [Knobbout and Dastani, 2012] should *still* be considered transition-based norms, even though they refer to certain actions of the agents. This is because in this work, it is asserted that a violation occurs in a run of the system if there is a transition from one state to another which can be caused by *some* forbidden action.

In the next section we present our logic, which allows us to reason about the strategies of agents under various compliance types.

3.3 Normative ATL

The goal of this section is to develop a logic in order to reason about the (normative) ability of agents, under assumption of various possible compliance types. But, before we can reason about *compliance* of agents, we have to define what we mean by this. This will be the topic of the next subsection. Afterwards, we will introduce the syntax and semantics of our logic, and prove several properties of this logic. We end this section with an elaborate example which shows how we can apply this logic.

3.3.1 Compliance types

In this chapter, we consider seven compliance types, particularly, three types of obedience, three types of disobedience, and one type of neglectfulness. We also introduce a special set of literals which we will use to identify these compliance types. As we will see later on in this chapter, if desired, our logical framework allows us to construct and reason about more compliance types as well. In other words, these seven compliance types are merely a *choice*, but they are arguably useful types to consider. Formally, they are defined as follows.

Definition 3.7 (Compliance types)**.** *A compliance type ω is an arbitrary literal $\omega \in \{c\text{-}ob, t\text{-}ob, s\text{-}ob, \top, s\text{-}dob, t\text{-}dob\} = \Omega$. Given a concurrent game structure $S = (Q, \Pi, \mu, Ags, act, \delta)$, norm set Γ, set of agents $A \subseteq Ags$ and run $r \in \mathcal{R}_S$, we say that if $A = \varnothing$, then r is (Γ, A, ω)-obedient for every $\omega \in \Omega$. If $A \neq \varnothing$, we say that r is ...*

1. *...* ***coalitional obedient****, or $(\Gamma, A, c\text{-}ob)$-obedient, iff at every position in r for every $A' \in \mathcal{P}_{\geq 1}(A)$ and every $\gamma \in \Gamma_{A'}$ it is the case that γ is not violated.*

2. *...* ***total individual obedient****, or $(\Gamma, A, t\text{-}ob)$-obedient, iff at every position in r for every $i \in A$ and every $\gamma \in \Gamma_{\{i\}}$ it is the case that γ is not violated.*

3. *...* ***selective individual obedient****, or $(\Gamma, A, s\text{-}ob)$-obedient, iff at every position in r there exists an $i \in A$ such that for every $\gamma \in \Gamma_{\{i\}}$ it is the case that γ is not violated.*

4. *...always* ***neglectful obedient****, or $(\Gamma, A, \top)$-obedient. Thus, every run is by definition neglectful obedient.*

5. *...* ***selective individual disobedient****, or $(\Gamma, A, s\text{-}dob)$-obedient, iff at every position in r there exists an $i \in A$ such that either (1) all $\gamma \in \Gamma_{\{i\}}$ are not enabled; or (2) there exists a $\gamma \in \Gamma_{\{i\}}$ such that γ is violated.*

6. *...* ***total individual disobedient****, or* $(\Gamma, A, t\text{-}dob)$*-obedient, iff at every position in* r *for every* $i \in A$ *it is the case that either (1) all* $\gamma \in \Gamma_{\{i\}}$ *are not enabled; or (2) there exists a* $\gamma \in \Gamma_{\{i\}}$ *such that* γ *is violated.*

7. *...* ***coalitional disobedient****, or* $(\Gamma, A, c\text{-}dob)$*-obedient, iff at every position in* r *for every* $A' \in \mathcal{P}_{\geq 1}(A)$ *it is the case that either (1) all* $\gamma \in \Gamma_{A'}$ *are not enabled; or (2) there exists a* $\gamma \in \Gamma_{A'}$ *such that* γ *is violated.*

In words, we say that a run is coalitional obedient with respect to A if all sub-coalitions of A always behave in accordance with the norms. Moreover, as a weaker variation we say that a run is total/selective individual obedient with respect to A if all/some agents in A behave in accordance with the norms. Additionally, we say that a run is coalitional disobedient with respect to A if all sub-coalitions of A always violate some norm if there is something to violate, and as a weaker variation we say that a run is total/selective individual disobedient if all/some agents always violate some norm if there is something to violate. It is thus important to note that for a disobedient run we only demand that a violation occurs if a violation *can* occur. We say that any run is neglectful obedient, since we are neglectful about whether a violation occurs. The following relations can be established between these obedience types.

Proposition 3.1. *Given a concurrent game structure* $S = (Q, \Pi, \mu, Ags, act, \delta)$*, a norm set* Γ*, a set of agents* $A \subseteq Ags$ *and a run* $r \in \mathcal{R}_S$*, for the obedience types we have:*

$$\begin{aligned} r \text{ is } (\Gamma, A, c\text{-}ob)\text{-}obedient &\Rightarrow r \text{ is } (\Gamma, A, t\text{-}ob)\text{-}obedient \\ r \text{ is } (\Gamma, A, t\text{-}ob)\text{-}obedient &\Rightarrow r \text{ is } (\Gamma, A, s\text{-}ob)\text{-}obedient \\ r \text{ is } (\Gamma, A, s\text{-}ob)\text{-}obedient &\Rightarrow r \text{ is } (\Gamma, A, \top)\text{-}obedient \end{aligned}$$

Additionally, for the disobedience types we have:

$$\begin{aligned} r \text{ is } (\Gamma, A, c\text{-}dob)\text{-}obedient &\Rightarrow r \text{ is } (\Gamma, A, t\text{-}dob)\text{-}obedient \\ r \text{ is } (\Gamma, A, t\text{-}dob)\text{-}obedient &\Rightarrow r \text{ is } (\Gamma, A, s\text{-}dob)\text{-}obedient \\ r \text{ is } (\Gamma, A, s\text{-}dob)\text{-}obedient &\Rightarrow r \text{ is } (\Gamma, A, \top)\text{-}obedient \end{aligned}$$

We can lift the notion of (dis)obedient runs to strategies as follows.

Definition 3.8 (Strategic compliance)**.** *Given a concurrent game structure* $S = (Q, \Pi, \mu, Ags, act, \delta)$*, a norm set* Γ*, a set of agents* $A \subseteq Ags$*, a state* q *and an compliance type* $\omega \in \Omega$*, we say that the strategy set* Σ_A *for* A *is* (Γ, A, ω)*-obedient if it holds that for every run* $r \in out_S(q, \Sigma_A)$ *we have that* r *is* (Γ, A, ω)*-obedient.*

As a remarkable consequence of the fact that we call a run disobedient whenever there are no norms to violate, whenever the norm set Γ is empty (and thus there is never a norm to violate), all possible strategies are both obedient *and* disobedient. Even though this may initially appear strange, we note that whenever Γ is not empty, the set of obedient strategies is different from the set of disobedient strategies.

With the compliance types that we defined in this section, we are ready to introduce our logic which allows us to reason about the strategic abilities of the agents in the presence of these types.

3.3.2 Syntax and semantics

In this subsection, we give the syntax and semantics of abstract normative Alternating-time Temporal Logic, or an-ATL, as introduced in [Knobbout and Dastani, 2012]. This logic allows us to reason about the strategic abilities of coalitions of agents under various compliance types, assuming that the remaining agents which are not part of this coalition behave in accordance with another compliance type. Formally, given a concurrent game structure $S = (Q, \Pi, \mu, Ags, act, \delta)$, an-ATL is characterized by the following grammar, where $p \in \Pi$, $A \subseteq Agt$ and $\omega, \omega' \in \Omega$:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\!\langle A_{(\omega,\omega')} \rangle\!\rangle \bigcirc \varphi \mid \langle\!\langle A_{(\omega,\omega')} \rangle\!\rangle \Box \varphi \mid \langle\!\langle A_{(\omega,\omega')} \rangle\!\rangle \varphi \mathcal{U} \varphi$$

Unlike ATL, a formula of this language is not only evaluated along a concurrent game structure and a state, but *also* along a norm set Γ. This is done in the following way.

Definition 3.9 (an-ATL semantics)**.** *Given a concurrent game structure $S = (Q, \Pi, \mu, Ags, act, \delta)$, a norm set Γ and a state $q \in Q$, the satisfaction relation $\vDash_{an\text{-}ATL}$ of an-ATL is inductively defined as follows:*

- $S, \Gamma, q \vDash_{an\text{-}ATL} p$ *iff* $p \in \mu(q)$.
- $S, \Gamma, q \vDash_{an\text{-}ATL} \neg\varphi$ *iff* $S, q \nvDash_{an\text{-}ATL} \varphi$.
- $S, \Gamma, q \vDash_{an\text{-}ATL} \varphi_1 \vee \varphi_2$ *iff* $S, q \vDash_{an\text{-}ATL} \varphi_1$ *or* $S, q \vDash_{an\text{-}ATL} \varphi_2$.
- $S, \Gamma, q \vDash_{an\text{-}ATL} \langle\!\langle A_{(\omega,\omega')} \rangle\!\rangle \bigcirc \varphi$ *iff there exists a* (Γ, A, ω)*-obedient strategy set* Σ_A *for* A *such that for every* $(\Gamma, Ags \backslash A, \omega')$*-obedient run* $r \in out_S(q, \Sigma_A)$ *it holds that* $S, \Gamma, r[1] \vDash_{an\text{-}ATL} \varphi$.
- $S, \Gamma, q \vDash_{an\text{-}ATL} \langle\!\langle A_{(\omega,\omega')} \rangle\!\rangle \Box \varphi$ *iff there exists a* (Γ, A, ω)*-obedient strategy set* Σ_A *for* A *such that for every* $(\Gamma, Ags \backslash A, \omega')$*-obedient run* $r \in out_S(q, \Sigma_A)$ *and all positions* $i \geq 0$ *it holds that* $S, \Gamma, r[i] \vDash_{an\text{-}ATL} \varphi$.

- $S, \Gamma, q \vDash_{an\text{-}ATL} \langle\!\langle A_{(\omega,\omega')} \rangle\!\rangle \varphi_1 \mathcal{U} \varphi_2$ *iff there exists a* (Γ, A, ω)*-obedient strategy set* Σ_A *for* A *such that for every* $(\Gamma, Ags \backslash A, \omega')$*-obedient run* $r \in out_S(q, \Sigma_A)$ *there exists a position* $i \geq 0$ *such that for all positions* $0 \leq j < i$ *it holds that* $S, \Gamma, r[j] \vDash_{an\text{-}ATL} \varphi_1$ *and* $S, \Gamma, r[i] \vDash_{an\text{-}ATL} \varphi_2$.

Whenever clear from context, we abbreviate $\vDash_{\text{an-ATL}}$ with simply $\vDash$. The formula $\langle\!\langle A_{(\omega,\omega')} \rangle\!\rangle \psi$ should intuitively be read as "coalition A has an (Γ, A, ω)-obedient strategy in order to enforce ψ if the remaining agents $Ags \backslash A$ played in accordance with an $(\Gamma, Ags \backslash A, \omega')$-obedient run". Similarly as in ATL, we write $[\![A_{(\omega,\omega')}]\!] \bigcirc \varphi$ for $\neg \langle\!\langle A_{(\omega,\omega')} \rangle\!\rangle \bigcirc \neg \varphi$ and $[\![A_{(\omega,\omega')}]\!] \Box \varphi$ for $\neg \langle\!\langle A_{(\omega,\omega')} \rangle\!\rangle \Diamond \neg \varphi$ (where $\Diamond \varphi \equiv \top \mathcal{U} \varphi$), and define similar abbreviations for the dual of the $\mathcal{U}$ operator. The formula $[\![A_{(\omega,\omega')}]\!] \psi$ should be read as "coalition A does not have a (Γ, A, ω)-obedient strategy in order to avoid ψ if the remaining agents played in accordance with an $(\Gamma, Ags \backslash A, \omega')$-obedient run". We additionally say that $S, \Gamma \vDash \varphi$ holds whenever for all $q \in Q$ it holds that $S, \Gamma, q \vDash \varphi$, and we say that $\vDash \varphi$ holds (alternatively, "φ is valid") whenever for all concurrent game structures S and norm sets Γ we have $S, \Gamma \vDash \varphi$.

Because this logic not only allows to reason about the strategic abilities of a coalition of agents, but also about the possible behaviour of the remaining agents, the things that we can assert using this logic are very expressive. As discussed in the introduction, this logic can be used to frame the verification objectives of a system designer. Later in this chapter, we will discuss model checking issues related to this logic. However, before we do, it is useful to look at certain properties and validities of this logic. After that, we will look at an extensive example which shows how this logic can be used to prove properties of a normative multi-agent system.

3.3.3 Properties and validities

An important property of an-ATL is that it is an extension of ATL. In particular, every ATL formula can be rewritten to a logically equivalent an-ATL formula.

Proposition 3.2 (ATL correspondence)**.** *Given an ATL formula* φ*, let* $\tau(\varphi)$ *be equal to* φ *except where every instance of* $\langle\!\langle A \rangle\!\rangle$ *is replaced with* $\langle\!\langle A_{(\top,\top)} \rangle\!\rangle$*. Given a concurrent game structure* $S = (Q, \Pi, \mu, Ags, act, \delta)$*, norm set* Γ *and state* $q \in Q$*, we have:*

$$S, q \vDash_{ATL} \varphi \quad \Leftrightarrow \quad S, \Gamma, q \vDash_{an\text{-}ATL} \tau(\varphi)$$

This proposition also highlights why the neglectfulness compliance type plays an important role, namely to ensure that our logic encapsulates standard ATL regardless of our choice for Γ. One of the most important properties of

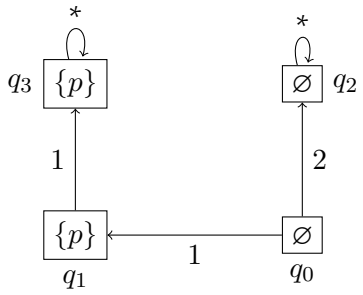


Figure 3.1: Concurrent game structure consisting of a single agent. Starting from state q_0, if the agent ought to avoid going from state q_1 to state q_3, there does not exist an obedient strategy for this agent to realize p in the next state.

an-ATL, as opposed to ATL, is its property of *non-locality*. In ATL, evaluating a formula of the form $\langle\!\langle A \rangle\!\rangle \bigcirc p$ would imply that we would only need to look at the next state in the model. In other words, the 'next' operator is a *local* temporal operator. In an-ATL, this operator behaves in a non-local way, and to see why this is the case we consider an example.

Example 3.1. *Consider the game structure S shown in Figure 3.1 consisting of a single agent. Moreover, consider the following norm set Γ:*

$$\Gamma = \{(q_1, \{1\}, q_3)\}$$

This norm states that agent 1 ought to avoid going from q_1 to q_3. Starting from q_0, agent 1 clearly has a strategy to realize p in the next state. We have the following:

$$S, \Gamma, q_0 \vDash \langle\!\langle \{1\}_{(\top,\top)} \rangle\!\rangle \bigcirc p$$

However, we do not have the following:

$$S, \Gamma, q_0 \nvDash \langle\!\langle \{1\}_{(t\text{-}ob,\top)} \rangle\!\rangle \bigcirc p$$

To see why this is not the case, even though agent 1 can go from q_0 to q_1 without violating any norms, he cannot do the same once he arrives in q_1. Particularly, once this agent arrives in q_1, he can do nothing else than to violate the norm. Thus, the strategy that goes from q_0 to q_1 ultimately leads to a violation, and is thus not a total individual obedient strategy.

This example highlights why our logic has the property of non-locality: in order to verify whether a strategy is (dis)obedient we also have to look at the possible future actions that may occur. Related to the property of non-locality is the property of (possible) non-existence of strategies, which we can again demonstrate with an example.

Example 3.2. *Consider again the game structure S from Figure 3.1 consisting of a single agent. We now consider the following norm set Γ:*

$$\Gamma = \{(q_0, \{1\}, q_1), (q_0, \{1\}, q_2)\}$$

Starting from q_0, agent 1 clearly has no total individual obedient strategy available to him, since every possible action causes him to violate a transition-based norm. The following holds:

$$S, \Gamma, q_0 \not\models \langle\!\langle \{1\}_{(t\text{-}ob,\top)} \rangle\!\rangle \bigcirc \top$$

Thus, although $\bigcirc\top$ holds for every possible run, there does not exist a strategy which is total individual obedient for agent 1, making the entire formula false.

This immediately illustrates an important difference from ATL, since ATL assumes that there is always a strategy available for any coalition of agents at any moment in time.

Strategic Relations

The logic of an-ATL is able to express many strategic relations of the agents. Particularly, we have the following validity, where specific notice should be laid on the reversal of the compliance types.

Proposition 3.3. *The following is valid for every formula ψ and compliance types $\omega, \omega' \in \Omega$:*

$$\models \langle\!\langle A_{(\omega,\omega')} \rangle\!\rangle \psi \rightarrow [\![(Ags \backslash A)_{(\omega',\omega)}]\!] \psi$$

Proof. Assume an arbitrary concurrent game structure S, norm set Γ and state q for which $S, \Gamma, q \models \langle\!\langle A_{(\omega,\omega')} \rangle\!\rangle \psi$. This implies there exists a (Γ, A, ω)-obedient strategy set such that for every $(\Gamma, Ags \backslash A, \omega')$-obedient run we have ψ. Let this strategy set be Σ_A. Now assume that $S, \Gamma, q \models \langle\!\langle (Ags \backslash A)_{(\omega',\omega)} \rangle\!\rangle \neg\psi$, i.e. there exists a $(\Gamma, Ags \backslash A, \omega')$-obedient strategy set such that for every (Γ, A, ω)-obedient run we have $\neg\psi$. Let this strategy set be $\Sigma_{Ags \backslash A}$. If we can derive that this leads to a contradiction, it must be the case that $S, \Gamma, q \not\models \langle\!\langle (Ags \backslash A)_{(\omega',\omega)} \rangle\!\rangle \neg\psi$, and thus $S, \Gamma, q \models [\![(Ags \backslash A)_{(\omega',\omega)}]\!] \psi$, as needed. From this assumption, we can conclude that there exists a run $r \in out_S(q, \Sigma_A) \cap out_S(q, \Sigma_{Ags \backslash A})$ which is both (Γ, A, ω)-obedient and $(\Gamma, Ags \backslash A, \omega')$-obedient, implying that r must satisfy ψ and $\neg\psi$. Contradiction! □

We also have the following validities, which shows an important relation between (dis)obedient strategies.

Proposition 3.4. *We have the following validities for every formula ψ and compliance type $\omega \in \Omega$:*

$$\models \langle\!\langle A_{(c\text{-}ob,\omega)} \rangle\!\rangle \psi \to \langle\!\langle A_{(t\text{-}ob,\omega)} \rangle\!\rangle \psi \quad \textit{and,} \quad \models \langle\!\langle A_{(c\text{-}dob,\omega)} \rangle\!\rangle \psi \to \langle\!\langle A_{(t\text{-}dob,\omega)} \rangle\!\rangle \psi$$
$$\models \langle\!\langle A_{(t\text{-}ob,\omega)} \rangle\!\rangle \psi \to \langle\!\langle A_{(s\text{-}ob,\omega)} \rangle\!\rangle \psi \quad \textit{and,} \quad \models \langle\!\langle A_{(t\text{-}dob,\omega)} \rangle\!\rangle \psi \to \langle\!\langle A_{(s\text{-}dob,\omega)} \rangle\!\rangle \psi$$
$$\models \langle\!\langle A_{(s\text{-}ob,\omega)} \rangle\!\rangle \psi \to \langle\!\langle A_{(\top,\omega)} \rangle\!\rangle \psi \quad \textit{and,} \quad \models \langle\!\langle A_{(s\text{-}dob,\omega)} \rangle\!\rangle \psi \to \langle\!\langle A_{(\top,\omega)} \rangle\!\rangle \psi$$

Intuitively, what these formulas state is that a coalition, when having an obedient strategy to guarantee ψ, they also have a "less restrictive" obedient strategy (where "less restrictive" means that there are less violations we have to consider) to also guarantee ψ (and vice versa for disobedient strategies). This is as expected, since the agents possibly have a richer pool of strategies to choose from when they have less restriction to cope with. This goes back to Proposition 3.1, which established these relations between compliance types and runs. Additionally, we can establish the following.

Proposition 3.5. *We have the following validities for every formula ψ and compliance type $\omega \in \Omega$:*

$$\models \langle\!\langle A_{(\omega,\top)} \rangle\!\rangle \psi \to \langle\!\langle A_{(\omega,s\text{-}ob)} \rangle\!\rangle \psi \quad \textit{and,} \quad \models \langle\!\langle A_{(\omega,\top)} \rangle\!\rangle \psi \to \langle\!\langle A_{(\omega,s\text{-}dob)} \rangle\!\rangle \psi$$
$$\models \langle\!\langle A_{(\omega,s\text{-}ob)} \rangle\!\rangle \psi \to \langle\!\langle A_{(\omega,t\text{-}ob)} \rangle\!\rangle \psi \quad \textit{and,} \quad \models \langle\!\langle A_{(\omega,s\text{-}dob)} \rangle\!\rangle \psi \to \langle\!\langle A_{(\omega,t\text{-}dob)} \rangle\!\rangle \psi$$
$$\models \langle\!\langle A_{(\omega,t\text{-}ob)} \rangle\!\rangle \psi \to \langle\!\langle A_{(\omega,c\text{-}ob)} \rangle\!\rangle \psi \quad \textit{and,} \quad \models \langle\!\langle A_{(\omega,t\text{-}dob)} \rangle\!\rangle \psi \to \langle\!\langle A_{(\omega,c\text{-}dob)} \rangle\!\rangle \psi$$

Intuitively, this means that a coalition, when having a certain strategy to guarantee ψ under a certain assumption that the other agents play in accordance with a certain obedient run, this strategy also guarantees ψ under assumption that the other agents play in accordance with a "more restrictive" obedient run (where "more restrictive" means that there are more violations we have to consider). Again, this goes back to Proposition 3.1, which established these relations between compliance types and runs.

In this section we discussed some properties and validities of our logic. Before we move to the problem of model checking a formula of an-ATL, it is useful to discuss an example which shows how this logic can be used to express properties of a normative multi-agent system. This will be the topic of the next section.

3.3.4 Example scenario

In this example we consider a scenario where there are two trains at the opposite ends of a tunnel, each controlled by an agent $i \in \{1, 2\}$. The tunnel only has room for one train, and the agents initially have 2 actions available, namely to wait and to go, which we will label with 'w' and 'go'. If the agents decide to go simultaneously, the trains will crash, if they wait simultaneously

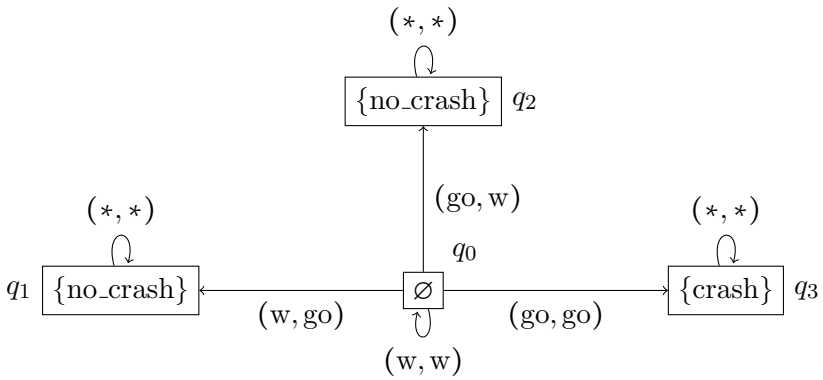


Figure 3.2: Concurrent game structure consisting of two agents controlling a train. Starting from initial state q_0 (middle bottom), both of them can either wait or go. If they both wait, nothing happens, and if they both go, a crash occurs. If one of them goes and one of them waits, no crash occurs.

nothing will happen and if one goes and the other waits, they can both successfully make it to the end of the tunnel without crashing. The concurrent game structure S belonging to this scenario is shown in Figure 3.2, where we use atomic propositions $\Pi = \{\text{crash}, \text{no\_crash}\}$ to denote whether a crash has occurred or not. A similar example with trains where the agents have to coordinate to avoid a crash can for example be found in [Woolridge and van der Hoek, 2005].

Immediately, we can establish the following, which states that none of the agents individually has the strategic ability to avoid a crash starting from q_0:

$$S, \varnothing, q_0 \not\models \langle\!\langle \{1\}_{(\top,\top)} \rangle\!\rangle \bigcirc \text{no\_crash}$$

And:

$$S, \varnothing, q_0 \not\models \langle\!\langle \{2\}_{(\top,\top)} \rangle\!\rangle \bigcirc \text{no\_crash}$$

Moreover, the agents are collectively able to make the trains crash, displayed by the following formula:

$$S, \varnothing, q_0 \models \langle\!\langle \{1,2\}_{(\top,\top)} \rangle\!\rangle \bigcirc \text{crash}$$

We construct a norm set Γ as follows:

$$\Gamma = \{(q_0, \{2\}, q_0), (q_0, \{2\}, q_2), (q_0, \{1,2\}, q_3)\}$$

In words, these transition-based norms state the following:

1. From state q_0, agent 2 ought to avoid transitioning to state q_0 or to state q_2.

2. From state q_0, agents 1 and 2 ought to avoid transitioning to state q_3.

In this example, we are first going to reason about the abilities of the first agent. The following an-ATL formula is valid:

$$S, \Gamma, q_0 \vDash \langle\!\langle 1_{(\top, t\text{-}ob)} \rangle\!\rangle \bigcirc \text{no\_crash}$$

In words, the first agent now has a strategy to bring about the fact that the train makes it through the tunnel without crashing under the assumption that the second agent plays according to a (total individual) obedient run. To see this, observe that we do not consider any runs with transitions from q_0 to q_0 or from q_0 to q_2. The strategy for the first agent is to wait, since he knows that second agent will not wait. Interestingly enough, the first agent also has a strategy to enforce this under the assumption that the second agent plays in accordance with a (total individual) *disobedient* run:

$$S, \Gamma, q_0 \vDash \langle\!\langle \{1\}_{(\top, t\text{-}dob)} \rangle\!\rangle \bigcirc \text{no\_crash}$$

To easily see this, observe that we do not consider any runs with transitions from q_0 to q_1 or from q_0 to q_3, thus the strategy where the first agent will immediately go ensures that no crash occurs.

Let us now reason about the abilities of the second agent. We can see that the second agent, as opposed to the first agent, cannot obediently bring about that both agents will reach the end of the tunnel safely:

$$S, \Gamma, q_0 \nvDash \langle\!\langle \{2\}_{(t\text{-}ob, \top)} \rangle\!\rangle \bigcirc \text{no\_crash}$$

To see this, selecting the action '*go*' in q_0 will not guarantee that we will not end up in q_1. However, interestingly enough, the second agent does have the ability to disobediently bring about the fact that the ability of the first agent to reach the end of the tunnel safely is not lost:

$$S, \Gamma, q_0 \vDash \langle\!\langle \{2\}_{(t\text{-}dob, \top)} \rangle\!\rangle \bigcirc \langle\!\langle \{1\}_{(\top, t\text{-}ob)} \rangle\!\rangle \bigcirc \text{no\_crash}$$

To see this, observe that the action 'wait' for the second agent in q_0 will result in either q_0 or q_2, both in which the ability of the first agent to safely reach the end of the tunnel, under assumption that the second agent behaves obediently, is retained. However, in case the second agent plays an obedient strategy, this is not guaranteed, as seen by the following:

$$S, \Gamma, q_0 \nvDash \langle\!\langle \{2\}_{(t\text{-}ob, \top)} \rangle\!\rangle \bigcirc \langle\!\langle \{1\}_{(\top, t\text{-}ob)} \rangle\!\rangle \bigcirc \text{no\_crash}$$

The reason for this is that the second agent cannot guarantee with a (total individual) obedient strategy (action 'go' in q_0) that he will not end up in q_3.

Let us finally reason about the abilities of the coalition of agents 1 and 2. We already saw that the coalition of agents have the ability to bring about the crashing of the trains. However, even if the individual agents play a totally individual obedient strategy (or a selective individual obedient strategy for that matter), the agents can still select a strategy that can make the trains crash, illustrated by the following validities:

$$S, \Gamma, q_0 \vDash \langle\!\langle \{1,2\}_{(t\text{-}ob,\top)} \rangle\!\rangle \bigcirc \text{crash}$$

and

$$S, \Gamma, q_0 \vDash \langle\!\langle \{1,2\}_{(s\text{-}ob,\top)} \rangle\!\rangle \bigcirc \text{crash}$$

The reason for this is because these compliance types apply only to the agents on an individual level, and thus we only have to take into account the norms associated with the second agent. On the coalitional level, the agents indeed do not have an coalitional obedient strategy to bring about the fact that a crash will occur, that is:

$$S, \Gamma, q_0 \nvDash \langle\!\langle \{1,2\}_{(c\text{-}ob,\top)} \rangle\!\rangle \bigcirc \text{crash}$$

Moreover, to illustrate how strong some compliance types can be, the following formula also holds:

$$S, \Gamma, q_0 \nvDash \langle\!\langle \{1,2\}_{(c\text{-}dob,\top)} \rangle\!\rangle \bigcirc \top$$

The reason for this is that there does not exist a strategy set for coalition $\{1,2\}$ which both violates a norm from $\Gamma_{\{2\}}$ and from $\Gamma_{\{1,2\}}$ (note that we do not have to consider $\Gamma_{\{1\}}$ because this set is empty).

In this section we considered an example scenario which shows how we can use our logic to express properties of a normative multi-agent system. However, we have not yet discussed the underlying complexity of deciding whether assertions from this language are true or false, which is the topic of model checking. How model checking can be done, and the associated complexity, will be the topic of the next section.

3.4 Model checking

This section is about the model checking problem for an-ATL. The model checking problem asks, given a game structure $S = (Q, \Pi, \mu, Ags, act, \delta)$, a norm set Γ and an an-ATL formula φ, for the set of states in $Q' \subseteq Q$ that satisfy φ. The model checking problem for general transition systems is discussed in [Clarke et al., 2001], and in [Alur et al., 2002] model checking for ATL is discussed.

Before we delve deeper into the problem, we introduce the notion of an *extended game structure*. As we will see, such a game structure allows us to identify (dis)obedient runs in the system.

Definition 3.10 (Extended game structure)**.** *Given a concurrent game structure* $S = (Q, \Pi, \mu, Ags, act, \delta)$ *and a norm set* Γ*, we define the extended game structure as* $S^F = (Q^F, \Pi^F, \mu^F, Ags, act^F, \delta^F)$*, such that:*

- $Q^F = \{(\bot, q) \mid q \in Q\} \cup$
  $\{(q', q) \mid q', q \in Q \text{ and } \exists \vec{\alpha} \in Act(q') : \delta(q', \vec{\alpha}) = q\}$
- $\Pi^F = \Pi \cup (\Gamma \times \{enabled, taken\})$
- $\mu^F((\bot, q)) = \mu(q)$*, and,*
  $\mu^F((q', q)) = \mu(q) \cup \{(\gamma, enabled) \mid \gamma = (q_{from}, A, q_{to}) \in \Gamma \text{ and } q_{from} = q'\}$
  $\cup \{(\gamma, taken) \mid \gamma = (q_{from}, A, q_{to}) \in \Gamma \text{ and } q_{to} = q\}$
- $act^F(i, (\bot, q)) = act^F(i, (q', q)) = act(i, q)$ *for every state* $(\bot, q), (q', q) \in Q^F$ *and every agent* $i \in Ags$*.*
- $\delta^F((\bot, q), \vec{\alpha}) = \delta^F((q', q), \vec{\alpha}) = (q, \delta(q, \vec{\alpha}))$ *for every state* $(\bot, q), (q', q) \in Q^F$ *and every joint action* $\vec{\alpha} \in Act(q)$*.*

There is a one-to-one correspondence between computations of S and S^F, and between strategies in S and S^F. The new propositions in $\Gamma \times \{\text{enabled}, \text{taken}\}$ allow us to identify the (dis)obedient runs. Note that if S contains m states, than S^F contains $O(m^2)$ states.

We will now show that, given an arbitrary coalition of agents A and norm set Γ, we can encode our compliance types using propositional logic. The language of propositional logic over a set of atomic propositions Π, denoted by $\mathcal{L}_{\text{prop}}(\Pi)$, is given by the following grammar, where $p \in \Pi$:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi$$

We define for a given norm $\gamma \in \Gamma$ the propositional formula $\text{violated}(\gamma) \in \mathcal{L}_{\text{prop}}(\Gamma \times \{\text{enabled}, \text{taken}\})$ as follows:

$$\text{violated}(\gamma) := (\langle \gamma, \text{enabled} \rangle \wedge \langle \gamma, \text{taken} \rangle)$$

Let f_A^Γ be a function that maps, given a coalition A and norm set Γ, each compliance type in Ω to a propositional formula with propositional atoms $\Gamma \times \{\text{enabled}, \text{taken}\}$. That is:

$$f_A^\Gamma : \Omega \to \mathcal{L}_{\text{prop}}(\Gamma \times \{\text{enabled}, \text{taken}\})$$

Whenever $A = \varnothing$, we let $f_A^\Gamma(\omega) = \top$. Whenever $A \neq \varnothing$, we let $f_A^\Gamma(\omega)$ be defined as follows (where $\bigvee_{\gamma \in \varnothing} = \bot$ and $\bigwedge_{\gamma \in \varnothing} = \top$):

1.
$$f_A^\Gamma(c\text{-}ob) = \bigwedge_{A' \in \mathcal{P}_{\geq 1}(A)} \left(\bigwedge_{\gamma \in \Gamma_{A'}} \neg \text{violated}(\gamma) \right)$$

2.
$$f_A^\Gamma(t\text{-}ob) = \bigwedge_{i \in A} \left(\bigwedge_{\gamma \in \Gamma_{\{i\}}} \neg \text{violated}(\gamma) \right)$$

3.
$$f_A^\Gamma(s\text{-}ob) = \bigvee_{i \in A} \left(\bigwedge_{\gamma \in \Gamma_{\{i\}}} \neg \text{violated}(\gamma) \right)$$

4.
$$f_A^\Gamma(\top) = \top$$

5.
$$f_A^\Gamma(s\text{-}dob) = \bigvee_{i \in A} \left(\bigvee_{\gamma \in \Gamma_{\{i\}}} \langle \gamma, \text{enabled} \rangle \rightarrow \bigvee_{\gamma \in \Gamma_{\{i\}}} \text{violated}(\gamma) \right)$$

6.
$$f_A^\Gamma(t\text{-}dob) = \bigwedge_{i \in A} \left(\bigvee_{\gamma \in \Gamma_{\{i\}}} \langle \gamma, \text{enabled} \rangle \rightarrow \bigvee_{\gamma \in \Gamma_{\{i\}}} \text{violated}(\gamma) \right)$$

7.
$$f_A^\Gamma(c\text{-}dob) = \bigwedge_{A' \in \mathcal{P}_{\geq 1}(A)} \left(\bigvee_{\gamma \in \Gamma_{A'}} \langle \gamma, \text{enabled} \rangle \rightarrow \bigvee_{\gamma \in \Gamma_{A'}} \text{violated}(\gamma) \right)$$

Given an arbitrary norm set Γ, observe that for every compliance type $\omega \in \Omega$ and coalition A we have that $f_A^\Gamma(\omega)$ is linear in the size of Γ. We have the result that evaluating a formula from the language of an-ATL exactly corresponds to evaluating a corresponding ATL* formula in the extended game structure. Particularly, we have the following correspondence.

Proposition 3.6 (Model checking correspondence)**.** *Given an an-ATL formula φ and norm set Γ, let $\tau(\varphi, \Gamma)$ be equal to φ except where every instance of $\langle\!\langle A_{(\omega,\omega')} \rangle\!\rangle \psi$ is replaced by:*

$$\langle\!\langle A \rangle\!\rangle (\Box(f_A^\Gamma(\omega)) \wedge (\Box(f_{(Ags \setminus A)}^\Gamma(\omega')) \rightarrow \psi))$$

Given an arbitrary concurrent game structure S, norm set Γ and state q, we have:

$$S, \Gamma, q \vDash_{an\text{-}ATL} \varphi \quad \Leftrightarrow \quad S^F, (\bot, q) \vDash_{ATL^*} \tau(\varphi, \Gamma)$$

Note that this proposition is aligned with the result we acquired in Proposition 3.2. Particularly, for an arbitrary norm set Γ, whenever we have an an-ATL formula φ where the only strategic modalities are of the form $\langle\!\langle A_{(\top,\top)} \rangle\!\rangle$, it is the case $\tau(\varphi, \Gamma)$ can be reduced to an ATL formula where every instance of $\langle\!\langle A_{(\top,\top)} \rangle\!\rangle$ is replaced by $\langle\!\langle A \rangle\!\rangle$. To illustrate, consider the formula $\langle\!\langle A_{(\top,\top)} \rangle\!\rangle \psi$:

$$\begin{array}{cc} \tau(\langle\!\langle A_{(\top,\top)} \rangle\!\rangle \psi, \Gamma) & \Leftrightarrow \\ \langle\!\langle A \rangle\!\rangle (\Box\top \wedge (\Box\top \rightarrow \tau(\psi, \Gamma))) & \Leftrightarrow \\ \langle\!\langle A \rangle\!\rangle \tau(\psi, \Gamma) & \end{array}$$

Although model checking an an-ATL formula equates to checking a logically equivalent ATL* formula, the model checking complexity still remains in the domain of polynomial time. This is an important result of this logic. We define the size of a concurrent game structure as the number of states and transitions that are present.

Theorem 3.1 (Model checking complexity)**.** *Given a game structure* $S = (Q, \Pi, \mu, Ags, act, \delta)$*, a norm set* Γ *and an an-ATL formula* φ*, the model checking problem for an-ATL is* **PTIME***-complete in the size of* S*, the size of* Γ*, and the length of* φ*.*

Proof. The proof strategy we adopt is based on the proof strategy found in [Alur et al., 2002, p695] to show **PTIME**-completeness of ATL. From S and Γ, we construct the extended game structure S^F as shown in Definition 3.10. From S^F we can construct a 2-player turn-based synchronous game S^F_A, where the first player controls all the actions of coalition A leading to an auxiliary state, after which the second player, controlling all the actions of coalition $Ags \backslash A$ decides on the next state. In this game, a move for coalition A is called an A-move, and a move for coalition $Ags \backslash A$ a B-move. Such a game can be interpreted as an AND-OR graph, for which we have to solve certain invariance and reachability problems, which can be done in efficient time as shown in [Beeri, 1980]. The interesting case happens when we need to calculate $[\langle\!\langle A_{(\omega,\omega')} \rangle\!\rangle \psi]$, where $[\varphi]$ denotes all the states that satisfy φ. We perform the following steps:

1. Remove all outgoing A-moves in S^F_A from a state that satisfies $\neg f^\Gamma_A(\omega)$, and repeat the following until we can do no more: Remove a state if it contains no out-going A-moves, and remove all auxiliary states which can reach this state with a B-move. Let $Q_1 \subseteq Q^F$ denote the non-auxiliary states which we remove in this step with this process. Here, Q_1 are the states for which there exists no (Γ, A, ω)-obedient strategy for A. The A-moves that remain are all aligned with a (Γ, A, ω)-obedient strategy.

2. Remove all outgoing B-moves in S_A^F from an auxiliary state that leads to a state that satisfies $\neg f^{\Gamma}_{(Ags \setminus A)}(\omega')$, and repeat the following until we can do no more: Remove an auxiliary state if it contains no out-going B-moves, and remove all states which can reach this state with an A-move. Let $Q_2 \subseteq Q^F$ denote the non-auxiliary states which we remove in this second step with this process. Here, Q_2 denote the states for which there exists a (Γ, A, ω)-obedient strategy Σ_A for A such that there does not exist a $r \in out_S(q, \Sigma_A)$ for which r is $(\Gamma, Ags \setminus A, \omega')$-obedient. The B-moves that remain are all aligned with a $(\Gamma, Ags \setminus A, \omega')$-obedient run.

3. Proceed with normal model checking in S_A^F, that is, calculate $Q_3 = [\langle\!\langle A \rangle\!\rangle \psi]$.

The procedure returns the set:

$$\{q \in Q \mid (\bot, q) \in \left((Q_3 \cup Q_2) \cap \overline{Q_1}\right)\}$$

Which are all the initial states of the original concurrent game structure (i.e. $(\bot, q)$ for a state $q \in Q$) for which the goal formula can be reached (Q_3), or for which there exists a (Γ, A, ω)-obedient strategy for A such that all possible runs are not $(\Gamma, Ags \setminus A, \omega')$-obedient ($Q_2$), and without all the states for which there does not exist a (Γ, A, ω)-obedient strategy for A (Q_1). Since every step in this procedure can be done in polynomial time (dependent on the size of the model, the length of the goal formula and the number of norms), membership to **PTIME** can be shown. To achieve the lower bound of **PTIME**-hardness, observe that due to Proposition 3.2 we have that ATL is encapsulated in an-ATL, for which hardness was already established. □

This concludes an important result of this chapter. Since we have developed this logic to perform formal verification on normative multi-agent system, it is worthwhile to discuss certain verification tasks which a designer might ask. This will be the topic of the next section.

3.5 Verification of norm sets

In this chapter we consider transition-based norms. Transition-based norms relate to certain transition that should be taken or avoided, as opposed to action-based norms which refer to certain actions that should be performed or neglected. An important question related to transition-based norms is whether agents have the strategic ability to avoid a violation. If this is not the case, we could say that these norms in some way restrict the 'autonomy' of the agents in the system, and thus can be considered as non-desired. Autonomy, in the broad sense, is a key facet within the agent paradigm and can play a

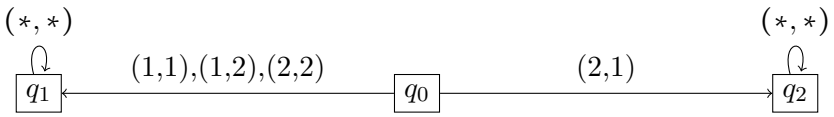


Figure 3.3: Starting from state q_0, both agents have two available actions they can perform (labelled 1 and 2). If it is the case that the first agent ought to avoid going from state q_0 to q_1, this second agent has a strategy to force the first agent into a violation.

major role for the agents to decide whether they want to participate in the multi-agent system or comply with the given norms, so we devote this last section to identify and classify the circumstances in which a norm set limits the autonomy of (coalitions of) agents. Let us consider an example.

Example 3.3. *We consider the concurrent game structure shown in Figure 3.3. Moreover, let us consider the norm set $\Gamma = \{(q_0, \{1\}, q_1)\}$, that is, agent 1 ought to avoid going from state q_0 to q_1. We see that the second agent, while being in state q_0, has the ability to enforce the first agent into a violation. Namely, we see that the second agent can select action 2, which causes the first agent to end up in q_1 regardless of the action he chooses.*

This brings us to the notion of self-supporting norm sets. Intuitively, self-supporting means that if a norm is targeted towards coalition A, the agents in this coalition have (in some way) "control" over whether or not they will violate this particular norm. However, as we will see in this section, multiple gradations of "control" can be given. We start out with a weak form of self-supporting, called *weakly self-supporting*.

Definition 3.11 (Weakly Self-supporting)**.** *Given a concurrent game structure S and norm set Γ, we say that Γ is weakly self-supporting with respect to S if and only if it holds that for every non-empty coalition $\varnothing \neq A \subseteq Ags$ and for every state $q \in Q$ there is no strategy available to A in order to force the remaining agents $Ags \backslash A$ into a non-$(\Gamma, Ags \backslash A, c\text{-}ob)$-obedient run.*

We see that in our previous example this was not the case since the second agent could force the first agent into a violation by selecting action 2. The following proposition shows how we can identify weakly self-supporting norm sets using our new an-ATL logic.

Proposition 3.7 (Weakly self-supporting)**.** *Given a concurrent game structure S and norm set Γ, Γ is weakly self-supporting with respect to S if and only if for all $\varnothing \neq A \subseteq Ags$ it holds that:*

$$S, \Gamma \vDash [\![A_{(\top, c\text{-}ob)}]\!] \bigcirc \top$$

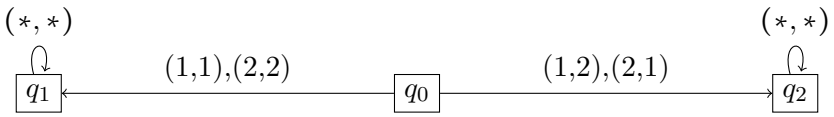


Figure 3.4: Starting from state q_0, both agents again have two available actions they can perform (1 and 2). If it is the case that the first agent ought to avoid going from state q_0 to q_1, this agent cannot choose a strategy to avoid a violation.

Proof. Assume an arbitrary $\varnothing \neq A \subseteq Ags$ for which $S, \Gamma \not\models [\![A_{(\top, c\text{-}ob)}]\!] \bigcirc \top$. We show that this holds iff Γ is not weakly self-supporting with respect to S. We have that $\not\models [\![A_{(\top, c\text{-}ob)}]\!] \bigcirc \top$ iff there exist a $q \in Q$ such that $S, \Gamma, q \models \langle\!\langle A_{(\top, c\text{-}ob)} \rangle\!\rangle \bigcirc \bot$. This holds iff there exist a strategy set Σ_A for A such that for all runs $r \in out_S(q, \Sigma_A)$ we have that r is not $(\Gamma, Ags \backslash A, c\text{-}ob)$-obedient (since $\bigcirc \bot$ holds for no possible run). This holds iff Γ is not weakly self-supporting with respect to S. □

To show that there exists stronger notions of self-supporting norm sets, consider the following example.

Example 3.4. *We consider the concurrent game structure shown in Figure 3.3, and again consider the norm set $\Gamma = \{(q_0, \{1\}, q_1)\}$, that is, agent 1 ought to avoid going from state q_0 to q_1. We see that Γ is weakly self-supporting with respect to S, since it is not possible for the second agent to force the first agent into a non-$(\Gamma, \{1\}, c\text{-}ob)$-obedient run. However, it is not the case that the first agent has a (coalitional) obedient strategy available to him in state q_0, as both actions 1 and 2 might bring him into state q_1.*

This brings up a stronger notion of self-supporting norm sets, namely those in which each coalition always has a coalitional obedient strategy available to them. If this is the case, we say that a norm set is *strongly self-supporting* with respect to a concurrent game structure.

Definition 3.12 (Strongly self-supporting)**.** *Given a concurrent game structure S and norm set Γ, we say that Γ is strongly self-supporting with respect to S if and only if it holds that for every non-empty coalition $\varnothing \neq A \subseteq Ags$ and for every state $q \in Q$ there is a coalitional obedient strategy available to A.*

Just like in the case of weakly self-supporting norm sets, we can identify when a norm set is strongly self-supporting with respect to a concurrent game structure with the use of our an-ATL logic.

Proposition 3.8 (Strongly self-supporting)**.** *Given a concurrent game structure S and norm set Γ, we say that Γ is strongly self-supporting with respect to S if and only if for all $A \subseteq Ags$ it holds that:*

$$S, \Gamma \vDash \langle\!\langle A_{(c\text{-}ob,\top)} \rangle\!\rangle \bigcirc \top$$

Proof. Follows trivially from the definition of $\langle\!\langle A_{(c\text{-}ob,\top)} \rangle\!\rangle \bigcirc \top$. □

We now claim that if a norm set is strongly self-supporting, it is also weakly self-supporting, and not the other way around as Example 3.4 shows. This is not hard to verify because, as we have already seen in Proposition 3.3, we have the following validity:

$$\vDash \langle\!\langle A_{(c\text{-}ob,\top)} \rangle\!\rangle \psi \rightarrow [\![(Ags \backslash A)_{(\top,c\text{-}ob)}]\!] \psi$$

Replacing ψ with $\bigcirc\top$, we establish the result, noting that $\langle\!\langle \varnothing_{(c\text{-}ob,\top)} \rangle\!\rangle \bigcirc \top$ and $[\![Ags_{(\top,c\text{-}ob)}]\!] \bigcirc \top$ are both tautologies within an-ATL.

What we have seen in this section is that it is possible to characterize and identify multiple levels of "control" the agents have over the ability to adhere to the norm set. As we already argued, identifying when a norm set is not (weakly/strongly) self-supporting with respect to a concurrent game structure may play a crucial role for the designer and the agents in the system, since they restrict the autonomy of the agents in some way.

3.6 Discussion

In this chapter we have developed a model of normative systems that allows reasoning of agents' (normative) abilities under a multitude of compliance types. This framework plays a crucial role in allowing us to perform formal verification on these systems. More specifically, we introduced the notion of transition-based norms and we developed an extension of Alternating-time Temporal Logic, an-ATL, to allow reasoning about the abilities of (coalitions of) agents under various compliance types. Moreover, we showed that the model checking complexity of an-ATL is equivalent to the model checking complexity of standard ATL. In the last part of the chapter, we discussed the notion of self-supporting norm sets and explained its relation to autonomy of agents. In particular, we explained that if a norm set is self-supporting, the agents have (in some way) control over whether they can avoid a violation. We have shown how an-ATL can be used to characterize these notions.

This chapter can be extended in multiple ways, which we will briefly reflect upon. Firstly, it would be very interesting to consider more compliance types. As we saw in Section 3.4, given a norm set Γ, our logic allows us to create arbitrary complex compliance types in the language $\mathcal{L}_{\text{prop}}(\Gamma \times \{\text{enabled}, \text{taken}\})$.

Moreover, it would be interesting to incorporate goals and preferences of agents and see how they relate to the obedience behaviour of the other agents. This will also be considered in the next chapter, where we assume that we can assign types to agents, where a type represents the knowledge and preferences of an agent. Lastly, it is important to consider more classes of regulative norms, for example state-based and sequence-based norms.

Chapter 4

Modelling and Verifying using Mechanism Design

In the previous chapter, we developed a logic which allowed us to reason about the strategic abilities of the agents under various sets of norms and under various compliance types of the agents. This approach allowed us to answer the verification problem of whether or not various (compliant or non-compliant) behaviours of the agents can lead to desired outcomes of the multi-agent system. However, this approach does not take into account *why* the agents would behave in such a manner. The agents might have personal preferences that decide how they will behave, and the approach in the previous chapter does not take these motives into account. Moreover, these preferences might not be known to the designer of the system. Once we make the assumption that the agents have *some* preference, and we are interested in how the agents *will* behave, we enter the field of game theory. Game theory, in the broad sense, is the study of strategic decision making in the presence of (one or more) rational agent(s), which has its roots in [Neumann and Morgenstern, 1944]. Since we also have norms that, once implemented, can change the environment of the system, these norms can act as a *mechanism* that can change the underlying game. If we are interested in whether we can design a norm such that the *predicted* behaviour (using game theory) coincides with the *desired* behaviour of the designer, we enter the field of mechanism design. In this chapter, we will frame our verification problem using principles from mechanism design, and we will provide a Unity-style calculus which can be used to solve our verification problem (Unity as introduced in [Chandy and Misra, 1989]). In the next section we will give a more detailed introduction to mechanism design, and show how concepts from mechanism design relate to our problem. At the end of that section, we give a detailed outline to the remainder of this chapter.

4.1 Introduction

In this chapter, we will frame our verification problem using principles from mechanism design. Mechanism design, and particularly implementation theory as a sub-component of this field, concerns itself with the following question: Given a goal related to the outcomes of a multi-agent system (alternatively *game*), can we design and verify a mechanism whose *predicted* outcomes coincide with the *desirable* outcomes, that is, the outcomes which are aligned with our goal? This immediately highlights a couple of important concepts, namely that of an *outcome*, that of a *mechanism*, the notion of *predicted outcomes*, and the notion of *desirable outcomes*. We will briefly go into more detail about these concepts. For a more general overview of Mechanism Design we direct the reader to [Osborne and Rubinstein, 1994], or for an overview that relates mechanism design to computer science, we direct the reader to [Nisan, 2007].

The outcomes of a system are considered the things we are interested in, and generally depend on the context. For example, in the context of an auctioneer selling assets, an outcome might be an allocation of assets across buyers, or in the context of an electorate seeking to fill a political office, an outcome is the choice of a candidate for that office (these examples were taken from Maskin's Nobel Memorial Prize Lecture in 2007, a reprinted version can be found in [Maskin, 2008]). In the context of normative systems, the outcomes we consider in this chapter are positive and negative sanctions we can assign to the agents. Often, these positive sanctions are referred to as rewards, and these negative sanctions as fines.

A mechanism is, in the general sense, an institution, procedure, protocol or game for generating outcomes. What a mechanism is, and the one who chooses the mechanism, again depends on the setting. In the example of an auctioneer selling goods, the mechanism is an auction protocol, and the one choosing the mechanism is the auctioneer. In the example an electorate seeking to fill a political office, the mechanism is an electoral procedure, for example a plurality rule, and the procedure is usually prescribed by the constitution. In the context of normative systems, the mechanism are the norms that are governed by the system in combination with a regulation policy. The ones we consider in this chapter are state-based norms, which tell the agents which states (situations) should be achieved or avoided by associating to these states certain sanctions. The designer of these mechanisms can be anything from a software engineer (in electronic systems) to a legislator.

The predicted outcomes are the outcomes we predict will occur, and we can derive these by using concepts from game theory. Because the mechanism designer generally does not know what the agents *know* and what the agents *value*, the predicted outcomes are conditioned on the *types* of the agents. A type is the combination of knowledge and preference of the agents, and a

designer usually considers a multitude of possible true types of the agents. In this chapter we do not wish to focus our attention on the knowledge that the agents possess, that is, we assume the agents possess common knowledge about the system and about their preferences. It is again very important to note that the designer of the mechanism does not have this knowledge, we only assume it is shared among the agents within the system. Whenever we consider a certain type profile for the agents, which is thus in our case the preference of each agent, in order to make a final prediction of the outcomes that will be achieved, we need certain rules that tell us which outcomes will be rationally optimal. In game theory, these formal rules are called the *solution concepts* that can be used for making these predictions. A multitude of these solution concepts exist, and the one that is most suited depends on a couple of factors. One important factor is the rationality constraints one might lay on the agents. Another important factor is that sometimes certain solution concepts may lead to multiple possible outcomes, which is often the case when considering weak equilibrium concepts. This puts any one of the solutions in doubt, so a game theorist may apply a refinement to narrow down the solutions, see e.g. [Osborne and Rubinstein, 1994]. In this chapter, we are using systems where agents act in sequence and we are using the concept of sub-game perfect Nash equilibria to characterize the predicted outcomes. This solution concept is a refinement of a Nash equilibrium used for sequential games.

The desired outcomes are the outcomes we want to have occur, and it again depends on the setting. In the context of an auctioneer selling assets, an important criterion might be that the assets are put into the hands of bidders who value them the most (that is, the allocation is *efficient*), and whether the greatest possible revenue is achieved for the seller (*revenue maximization* is achieved). In the example of an electorate seeking to fill a political office, the preferred candidate would be the one who, when compared with every other candidate, is preferred by most voters (the *Condorcet candidate*). This is the domain of *social choice theory*, which concerns itself with combining individual preference in order to reach social welfare [Arrow, 1951]. A lot of work has already been spent on trying to find suitable criteria for optimal outcomes, and a lot of work exists on showing when this is impossible, most famously Arrow's impossibility theorem [Arrow, 1950]. In the context of normative systems, the desired outcomes are the ones that maintain order in society. For example, a typical criterion one may adopt in a utilitarian society is that wrong-doers should be punished. On the other hand, punishment has consequences for both the offender and society. Thus, another criterion might be that somebody should only be punished if the total good produced by the punishment should exceed the total evil. These are all very complicated issues, and we do not wish to focus our attention on these matters too much in this work. We simply

assume that such rules are already decided upon (depending on the kind of society we want to consider), and this knowledge is compressed into a single social-welfare rule.

Because mechanism designers do not know the agent types beforehand, a more cautious approach has to be employed. For example in a normative multi-agent system, we do not know beforehand if an agent has the incentive of committing a crime, and as such, we cannot simply prescribe sanctions before any execution has occurred. This information must be generated as the system is executed. The problem here is the fact that the agents in the system may have their own objectives, and may try to behave in a way that hides the truth. A typical goal of a mechanism designer is thus often to develop mechanisms that are incentive compatible, i.e. the mechanism is capable of revealing the true incentives of the agents. This sets the basic playing field of mechanism design and implementation theory: Can we design and verify a mechanism whose *predicted* outcomes coincide with the *desirable* outcomes? This question can be approached from a multitude of possible directions. In this chapter we first frame our verification problem using these principles from mechanism design, and secondly provide a Unity-style calculus which can be used to solve our verification problem.

This work is related to the work of [Bulling and Dastani, 2011b], which also studies mechanism design in which the norms act as the underlying mechanism. This work differs in a multitude of ways, but two noteworthy differences are that (1) the outcomes they consider are (infinite) runs of the system, while the outcomes we consider are the positive and negative sanctions we can assign to the agents, and most importantly (2) we give logical characterizations of the implementability questions and try to answer these question with the use of a logical proof system. In other words, in this chapter we will concern ourselves with *deductive verification*. These logical characterizations of the implementability questions are inspired by the work of [Pauly, 2002], where they use an extension of Hoare logic to give similar characterizations. This work fundamentally differs from this work in the sense that we are interested in the verification of normative multi-agent systems, and not in imperatively specified mechanisms and protocols.

4.1.1 The verification problem

The verification problem we consider in this chapter is the following: Given the specification of a multi-agent system, a set of possible agent types, a social choice rule denoting what is optimal for society, and a norm acting as the underling mechanism, can we verify whether this mechanism implements the social choice rule in sub-game perfect equilibrium. The corresponding synthesis and design question is whether, and to which extent, we can design

such a mechanism.

4.1.2 Chapter outline

We give a brief outline of the contents of this chapter:

- In **Section 4.2** (p 57) we introduce the model of execution we are interested in, define the notion of sequence-based and state-based norms, and define how our model of execution in the presence of such norms can be viewed as a game.
- In **Section 4.3** (p 72) we frame our verification problem using the principles from mechanism design. This section ends by showing an extensive example, which we will return to at the end of the chapter.
- In **Section 4.4** (p 80) we introduce a Unity-style proof system that allows us to infer game-theoretic properties of multi-agent systems.
- In **Section 4.5** (p 101) we show how we can apply this proof system to show when a norm implements a social choice rule in a multi-agent system.
- In **Section 4.6** (p 108) we discuss this chapter.

4.2 Preliminaries

In Chapter 3, we used concurrent game structures as a model for multi-agent systems. These structures assume that there exists a transition relation between states specifying how we can transition from one state to another state with the use of a (joint) action. Typically these models are not concerned with the structure of an action, i.e. what the preconditions and postconditions of an action are. In this chapter we want to reason about how an individual action of an agent affects (the state of) the multi-agent system, thus we want to have an explicit notion of an action. In planning problems, like STRIPS instances, this is typically done by assuming a set of actions specified in terms of pre- and postcondition, and a set of basic facts on which these actions operate. We use the same idea in this chapter, where we assume we have a set of atomic propositions (facts) Π, and a set of actions such that the precondition of each action is a propositional formula ranging over these facts. In Chapter 3 we already briefly introduced the logical language of propositional logic, but we will repeat it here since it will play a very important role in this chapter. The language of propositional logic over a set of atomic propositions Π, denoted by $\mathcal{L}_{\text{prop}}(\Pi)$, is given by the following grammar, where $p \in \Pi$:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi$$

A formula from this language is evaluated along a set $\Pi' \subseteq \Pi$. To avoid confusion within this chapter, we write $\models_{\text{prop}}$ to explicitly refer to the satisfaction relation of propositional logic, and is defined as follows.

Definition 4.1 (Propositional Logic semantics)**.** *Given a set of atomic propositions Π', the satisfaction relation $\models_{prop}$ of propositional logic is inductively defined as follows:*

- $\Pi' \models_{prop} p$ *iff* $p \in \Pi'$.
- $\Pi' \models_{prop} \neg\varphi$ *iff* $\Pi' \not\models_{prop} \varphi$.
- $\Pi' \models_{prop} \varphi_1 \vee \varphi_2$ *iff* $\Pi' \models_{prop} \varphi_1$ *or* $\Pi' \models_{prop} \varphi_2$.

In the remainder of this section, we introduce the model of execution we are interested in, define our notion of norms, and define how our model of execution in the presence of such norms can be viewed as a game.

4.2.1 Turn-based multi-agent systems

The model of execution we want to consider has the following characteristics. We assume that there exists a non-empty set of agents, and one of the agents, depending on the current state we are in (starting from some initial state), gets to decide an action from the set of applicable actions. This action then deterministically progresses the current state to the next state, at which point this process repeats. It is thus important to note that we do not consider joint actions in this chapter, we simply assume that at each state exactly one agent gets to decide the next state. Moreover, we also assume that on top of the facts which are changed by actions, which we refer to as hard facts, there also exist soft facts which encode the (positive or negative) sanctions the agents may receive. These facts will play an important role later when we are going to talk about norms. Formally, *turn-based multi-agent systems* are defined as follows.

Definition 4.2 (Turn-based multi-agent systems)**.** *A turn-based synchronous multi-agent systems is a structure $M = (\Pi, Ags, ag, Act, H_0)$ such that:*

- $\Pi = \Pi_{\text{hard}} \cup \Pi_{\text{soft}}$ *is a finite set of propositions such that* $\Pi_{\text{hard}} \cap \Pi_{\text{soft}} = \varnothing$. *We call a subset of atomic hard propositions $H \in \mathcal{P}(\Pi_{\text{hard}})$ a* hard state *and a subset of atomic soft propositions $S \in \mathcal{P}(\Pi_{\text{soft}})$ a* soft state.
- $Ags = \{1, \ldots, n\}$ *is a finite non-empty set of agents.*
- $ag : \mathcal{P}(\Pi_{\text{hard}}) \to Ags$ *is a function that assigns to each hard state $H \in \mathcal{P}(\Pi_{\text{hard}})$ an agent $ag(H) \in Ags$ that gets to decide an action.*

- *Act is a finite set of actions. An action α has the form $(\varphi, \Pi^+_{\text{hard}}, \Pi^-_{\text{hard}})$, where $\varphi \in \mathcal{L}_{prop}(\Pi_{\text{hard}})$ is* the precondition *, and $\Pi^+_{\text{hard}}, \Pi^-_{\text{hard}} \subseteq \Pi_{\text{hard}}$ (for which $\Pi^+_{\text{hard}} \cap \Pi^-_{\text{hard}} = \varnothing$) the hard facts which become true, respectively false, after execution of this action.*

- *$H_0 \in \mathcal{P}(\Pi_{\text{hard}})$ is the initial hard state of the system.*

Given a hard state $H \in \mathcal{P}(\Pi_{\text{hard}})$, we write Act_H to denote the set $Act_H = \{(\varphi, \Pi^+_{\text{hard}}, \Pi^-_{\text{hard}}) \in Act \mid H \vDash_{prop} \varphi\}$, which is the set of applicable actions in H. Given an action $\alpha = (\varphi, \Pi^+_{\text{hard}}, \Pi^-_{\text{hard}}) \in Act_H$ we write $H(\alpha)$ to denote the uniquely defined updated hard state $(H \cup \Pi^+_{\text{hard}}) \backslash \Pi^-_{\text{hard}}$. Given a multi-agent system M and hard state $H \in \mathcal{P}(\Pi_{\text{hard}})$, we write (M, H) to denote the multi-agent system in which H is the initial hard state.

From here-on in this chapter, we refer to a structure M simply as a multi-agent system instead of a turn-based multi-agent system. Moreover, in the context of such a system, whenever we refer to a set H we implicitly refer to a hard state from the set $\mathcal{P}(\Pi_{\text{hard}})$, and whenever we refer to a set S, we implicitly refer to a soft state from the set $\mathcal{P}(\Pi_{\text{soft}})$. As is clear from the above definition, actions only operate on hard states, are performed by a single agent at a given state, and deterministically progress a state to a next state. Relating to the various properties of multi-agent systems we discussed in Chapter 2, this model is thus turn-based, synchronous, decentralized, discrete and deterministic. Note that in this model it can occur that for a given state H, the set Act_H is empty, i.e. there exist no applicable actions for agent $ag(H)$. In this case, we say that the execution of a multi-agent system ends, or alternatively, the execution arrives into a deadlock state.

A multi-agent system gives rise to a set of possible behaviours. In the previous chapter, we were mainly interested in the possible computations (infinite sequences) that could arise when running such a system. In this chapter, we are interested in the possible histories (finite sequences) that can occur when running the system. The set of histories is defined as follows.

Definition 4.3 (Histories)**.** *Given a multi-agent system $M = (\Pi, Ags, ag, Act, H_0)$, a history in M is a finite non-empty sequence of hard states $H_0 \dots H_n$ starting from initial hard state H_0 such that for each position t $(0 \leq t < n)$ it holds that $\exists \alpha \in Act_{H_t}$ such that $H_t(\alpha) = H_{t+1}$. The last occurring state of a history h is denoted by H_h. The set of all histories given M is denoted by $\mathcal{H}_M$. Given two histories $h, h' \in \mathcal{H}_M$, we let $h \sqsubseteq h'$ denote that h is a prefix of h', and $h \sqsubset h'$ that h is a proper prefix of h' ($h \sqsubseteq h'$ and $h \neq h'$). The length of a history h , notation $|h|$, is defined as the number of actions that are performed along such a history, i.e. $|H_0 \dots H_n| = n$. Given an arbitrary history $h = H_0 \dots H_n \in \mathcal{H}_{(M,H_0)}$ and $h' = H_n \dots H_m \in \mathcal{H}_{(M,H_n)}$, we denote*

with $(h+h')$ *the concatenated history* $H_0 \dots H_n \dots H_m$*, for which we have that* $|h + h'| = |h| + |h'|$.

The set of histories given a multi-agent system may be a finite or infinite set, depending on the applicable actions that exists along the states of the system. Note that there should be no confusion between the syntactically similar elements H (a hard state), h (a history, which is a sequence of hard states) and $\mathcal{H}_M$ (set of all possible histories in system M). Also note that we have to be very careful when concatenating histories, since it is easy to make a mistake. Given a history H_0H_1 ($|H_0H_1| = 1$) and H_1H_2 ($|H_1H_2| = 1$), we say that $(H_0H_1 + H_1H_2) = H_0H_1H_2$ ($|H_0H_1H_2| = 2$), and not $H_0H_1H_1H_2$. As we will see later on in this chapter, this notation helps us to define how we can combine histories to create larger histories of the system.

In the next section we will discuss the kinds of norms we consider in this chapter, and how the introduction of these norms to the system can change the histories of the system.

4.2.2 Sequence-based norms

The kinds of norms we consider in this chapter are *sequence-based norms*, which refer to certain good and bad histories of a system. A very abstract way to model such norms is to view them as functions that take a possible history of the system, and output a set of soft facts that ought to be the case after such a history has occurred. Such a history can then be considered 'good behaviour' if this set contains merely positive sanctions (rewards) for the agents and can be considered 'bad behaviour' if this set contains merely negative sanctions (fines) for the agents, but in general such a set can contain both positive and negative sanctions for the agents. Moreover, we consider the possibility that such a norm does not output a set of soft facts for every history, and can leave certain histories *undetermined*. To give a basic intuition as to what this means, imagine a norm on the motorway stating that the average speed of every car along a certain section should not exceed a certain limit. Now, only after a car has left this section we can decide whether the agent controlling the car should be given a fine or not. When this car is still on the section, this value of this fine is still undetermined. In other words, the evaluation of a certain behaviour does not have to occur at every moment along a history, and can only occur along certain pre-determined histories. As such, a sequence-based norm can be compactly represented by a single partial function that *possibly* assigns to a history of a system a soft state that should be the case after this history has occurred. Formally, they are defined as follows.

Definition 4.4 (Sequence-based norms, assigned soft states)**.** *Given a multi-*

agent system $M = (\Pi, Ags, ag, Act, H_0)$*, a sequence-based norm* f *is a* partial function *which possibly maps a history from* $\mathcal{H}_M$ *to a soft state* $\mathcal{P}(\Pi_{\text{soft}})$*. The domain of a sequence-based norm* f *is denoted by* $dom_M(f) \subseteq \mathcal{H}_M$*. Whenever for a history* $h \in \mathcal{H}_M$ *we have that* $h \in dom(f)$*, we say that history* h *is* assigned *soft state* $f(h)$*, otherwise we say that* h *is not* assigned *a soft state. Likewise, given a formula* $\psi \in \mathcal{L}_{prop}(\Pi)$*, we define the domain of a formula* $dom_M(\psi)$ *as all the histories which end in a hard state for which there exists a soft state which satisfy this formula. That is:*

$$dom_M(\psi) := \{h \in \mathcal{H}_M \mid \exists S \in \mathcal{P}(\Pi_{\text{soft}}) : H_h \cup S \vDash_{prop} \psi\}$$

Any formula $\psi \in \mathcal{L}_{prop}(\Pi)$ *encodes a (non-empty) set of alternative sequence-based norms,* $\mathcal{N}_M(\psi)$*, as follows:*

$$\mathcal{N}_M(\psi) := \{f \in [dom_M(\psi) \to \mathcal{P}(\Pi_{\text{soft}})] \mid \forall h \in dom_M(\psi) : (H_h \cup f(h) \vDash_{prop} \psi)\}$$

Whenever the set $\mathcal{N}_M(\psi)$ *is a singleton, i.e.* $\mathcal{N}_M(\psi) = \{f\}$*, we call the formula* ψ functional*.*

In this chapter, we impose no structural restrictions on such a function. In a more specific setting we might for example want to have a property that a sanction lasts until a repair action has been performed, at which point the sanction disappears. These more structurally focussed norms will be the subject of Chapter 5, where we will discuss the topic of dynamic normative systems. A sequence-based norm can be *implemented* in a normative system with the use of some enforcement mechanism. In this thesis, we do not want to focus our attention on how such a mechanism works, we are merely interested in the effect of this implementation. The next definition shows what the values of the soft facts become if a sequence-based norm is implemented in a normative system.

Definition 4.5 (Implemented sequence-based norms)**.** *Given a multi-agent system* $M = (\Pi, Ags, ag, Act, H_0)$*, history* $h \in \mathcal{H}_M$ *and soft fact* $o \in \Pi_{\text{soft}}$*, we say that a sequence-based norm* f *is* implemented *in* M *if and only if:*

- o *is* true *after the occurrence of* h *if* $h \in dom(f)$ *and* $o \in f(h)$*, and;*
- o *is* false *after the occurrence of* h *if* $h \in dom(f)$ *and* $o \notin f(h)$*, and;*
- o *is* undetermined *(alternatively* unassigned*) after the occurrence of* h *if* $h \notin dom(f)$*.*

From this definition it is clear that when the empty function (the norm whose domain is empty) is implemented, it leaves the behaviour of the system unchanged, since this implies that no soft fact is ever assigned.

How a propositional formula can represent a set of alternative sequence-based norms perhaps requires a bit more explanation, and will we do this with the use of the following example.

Example 4.1. *Suppose we have a multi-agent power plant system M consisting of two generators, which can be powered on or off. We use hard fact g_0 to denote that the first generator is on, and use hard fact g_1 for the second generator. Thus $\Pi_{\text{hard}} = \{g_0, g_1\}$. Moreover, the system uses violations atoms to label bad situations that can occur, for which we use $\Pi_{\text{soft}} = \{v_0, v_1\}$. We assume that the system can generate the following set of histories $\mathcal{H}_M = \{h_0, h_1, h_2\}$ such that:*

- $h_0 = \{\}$
- $h_1 = \{\}\{g_0\}$
- $h_2 = \{\}\{g_0\}\{g_1\}$

Note that throughout this chapter we write a history as a non-comma separated list of states, although mathematically they represent vectors over states. The set $\mathcal{N}_M(\neg g_0 \wedge v_0 \wedge \neg v_1) = \{f_0\}$ contains a single sequence-based norm f_0 in which every history that ends in a hard state where the first generator is powered off $(\neg g_0)$ is assigned violation v_0 and not violation v_1, that is, the soft state $\{v_0\}$. Thus note that a soft state contains only the soft facts which are true, while the soft facts that are false are left out. Formally, we have:

$$dom(\neg g_0 \wedge v_0 \wedge \neg v_1) = dom(f_0) = \{h_0, h_2\}$$

And:

$$f_0(h_0) = f_0(h_2) = \{v_0\}$$

As a second example, we consider the set $\mathcal{N}_M(\neg g_1 \wedge v_1) = \{f'_0, f'_1, f'_2, f'_3\}$, which contains four distinct sequence-based norms. To see why this is the case, consider that any history ending in a hard state where the second generator is off $(\neg g_1)$ can be either assigned soft state $\{v_1\}$ or soft state $\{v_0, v_1\}$. The reason why the latter soft state is also a possibility is because the propositional formula $\neg g_1 \wedge v_1$ says nothing about the truth of v_0, so both soft states are considered. We thus have:

$$dom(\neg g_1 \wedge v_1) = dom(f'_0) = dom(f'_1) = dom(f'_2) = dom(f'_3) = \{h_0, h_1\}$$

And:

$$\begin{aligned} f'_0(h_0) = \{v_1\}, \quad & f'_0(h_1) = \{v_1\} \\ f'_1(h_0) = \{v_1\}, \quad & f'_1(h_1) = \{v_0, v_1\} \\ f'_2(h_0) = \{v_0, v_1\}, \quad & f'_2(h_1) = \{v_1\} \\ f'_3(h_0) = \{v_0, v_1\}, \quad & f'_3(h_1) = \{v_0, v_1\} \end{aligned}$$

It is important to note that the set of sequence-based norms encoded by a formula is always non-empty, and if the set of possible histories is infinite, this set of possible sequence-based norms can also be infinite. Even for the formula $\bot$ (representing a propositional contradiction) we have for a given multi-agent system M that the set $\mathcal{N}_M(\bot)$ is non-empty, namely it consists of the empty norm (the norm whose domain is empty) since $dom_M(\bot) = \varnothing$. This norm leaves the behaviour of any system unchanged. In this chapter, we are particularly interested in a subset of these sequence-based norms which only differentiate between good and bad *hard states* of the system, and not between arbitrary-length histories. These *state-based norms* are discussed in the next section.

State-based norms

In this chapter, we define state-based norms as a special kind of sequence-based norm with the property that this norm maps every history ending in the same hard state to the same soft state. Formally they are defined as follows.

Definition 4.6 (State-based norms)**.** *Given system M, a sequence-based norm f is a state-based norm if and only if for all $h \in dom_M(f)$ and $h' \in \mathcal{H}_M$ we have $(H_h = H_{h'}) \Rightarrow (h' \in dom_M(f)$ and $f(h) = f(h'))$.*

Whenever a formula is functional, this formula always encodes a state-based norm.

Proposition 4.1. *Given multi-agent system $M = (\Pi, Ags, ag, Act, H_0)$ and a sequence-based norm f, if there exists a functional formula ψ such that $\mathcal{N}_M(\psi) = \{f\}$, then f is a state-based norm.*

Proof. Given sequence-based norm f, let ψ be the functional formula for which we have that $\mathcal{N}_M(\psi) = \{f\}$. From this assumption we can immediately conclude that $dom_M(\psi) = dom_M(f)$. Let $h \in dom_M(f)$ and $h' \in \mathcal{H}_M$ be two arbitrary histories for which we have that $H_h = H_{h'}$. Since $h \in dom_M(f)$ and thus $h \in dom_M(\psi)$, we have that $H_h \cup f(h) \vDash_{\text{prop}} \psi$. Since $H_h = H_{h'}$, we thus also have that $H_{h'} \cup f(h) \vDash_{\text{prop}} \psi$, and thus $h' \in dom(\psi)$, implying that $h' \in dom(f)$. However, since ψ is functional, it must mean that there exists exactly one $S \in \mathcal{P}(\Pi_{\text{soft}})$ such that $H_{h'} \cup S \vDash_{\text{prop}} \psi$, which implies that $S = f(h) = f(h')$, as needed. Because h and h' were chosen arbitrarily, it must hold that for all $h \in dom_M(f)$ and $h' \in \mathcal{H}_M$ we have $(H_h = H_{h'}) \Rightarrow (f(h) = f(h'))$, as needed. □

The other way around, i.e. that every state-based norm can be encoded by a functional formula, is also true. Before we show that this is the case, we introduce some notation which we not only use in the proof of the upcoming

proposition, but which we use throughout the chapter. That is, whenever we write ϕ_H^{hard} (or ϕ_S^{soft}) in the context of a set Π ($\Pi = \Pi_{\text{soft}} \cup \Pi_{\text{hard}}$) within a *propositional formula*, we mean the following:

$$\phi_H^{\text{hard}} := \left(\bigwedge_{p \in H} p\right) \wedge \left(\bigwedge_{p \in (\Pi_{\text{hard}} \setminus H)} \neg p\right), \text{ and, } \phi_S^{\text{soft}} := \left(\bigwedge_{p \in S} p\right) \wedge \left(\bigwedge_{p \in (\Pi_{\text{soft}} \setminus S)} \neg p\right)$$

Given a hard state H, we have that $H \vDash_{\text{prop}} \phi_H^{\text{hard}}$ and there exists no hard state $H' \neq H$ such that $H' \vDash_{\text{prop}} \phi_H^{\text{hard}}$. Likewise, given a soft state S, $S \vDash_{\text{prop}} \phi_S^{\text{soft}}$ and there exists no soft state $S' \neq S$ such that $S' \vDash_{\text{prop}} \phi_S^{\text{soft}}$. These formulas will be used throughout this chapter whenever we want to refer to a particular hard or soft state within a logical formula. We have the following result, which shows that any possible state-based norm can be encoded by a propositional formula.

Proposition 4.2. *Given a multi-agent system $M = (\Pi, Ags, ag, Act, H_0)$ and a sequence-based norm f, if f is a state-based norm then there exists a functional formula ψ such that $\mathcal{N}_M(\psi) = \{f\}$.*

Proof. Given an arbitrary state-based norm f, let:

$$R = \{(H_h, f(h)) \mid h \in dom_M(f)\}$$

Here, R denotes a functional relation consisting of all the hard-soft state pairs (H, S) for which f assigns to hard state H soft state S. This relation is finite, since the set of all possible states is finite, and is functional, since f is state-based. Now we let ψ be defined as follows:

$$\psi = \bigvee_{(H,S) \in R} (\phi_H^{\text{hard}} \wedge \phi_S^{\text{soft}})$$

For this particular ψ it holds that for every history h and soft state S that $H_h \cup S \vDash_{\text{prop}} \psi$ iff $f(h) = S$, which implies that $\mathcal{N}_M(\psi) = \{f\}$, as needed. □

The result of this proposition is that there is a direct correspondence between state-based norms and functional formulas. Thus, instead of representing state-based norms as functions, we can also uniquely represent state-based norms as (functional) formulas. In the remainder of this chapter, we will use formulas to represent state-based norms, and we will call them as such.

Definition 4.7 (Logical representation of state-based norms)**.** *Given a multi-agent system M, we call a propositional formula $\psi \in \mathcal{L}_{prop}(\Pi)$ a state-based norm iff ψ is functional.*

In this section we have shown how we can represent sequence-based and state-based norms within this framework. These norms can be used to steer an agent towards (or away) from certain states. How this process works is described in the next section, where we develop a framework where we can view the execution of a multi-agent system as a finite game that the agents are playing.

4.2.3 Finite extensive games

In this section, we develop a framework which allows us to view the possible executions of a multi-agent system in the presence of a sequence-based norm as a game. In a normative multi-agent system, the soft facts are generally understood as (positive or negative) sanctions the agents can receive, such as a fine. In this context, we can assume that every agent has certain preferences over (sub)sets of these soft facts (the soft states), which the designer of the system may not know in advance. Formally, *preferences* are defined as follows.

Definition 4.8 (Preferences)**.** *Given a multi-agent system* $M = (\Pi, Ags, ag, Act, H_0)$*, a preference of an agent* $i \in Ags$*, denoted by* $\succsim_i$*, is a complete reflexive transitive binary relation over soft states from* $\mathcal{P}(\Pi_{\text{soft}})$*. If for two soft states* $S, S' \in \mathcal{P}(\Pi_{\text{soft}})$ *it holds that* $S \succsim_i S'$ *and* $S' \succsim_i S$*, we write* $S \sim_i S'$*, and when* $S \succsim_i S'$ *and* $S' \not\succsim_i S$*, we write* $S \succ_i S'$*. A preference profile* $\succsim = (\succsim_1, \ldots, \succsim_{|Ags|})$ *consists of a preference of each agent. We call a multi-agent system* M *together with a preference profile* $\succsim$ *a multi-agent preference structure, and we denote it with* $M_{\succsim}$*.*

An extensive-form game is a general way to represent games in which there exists a sequencing of moves. Broadly speaking, an extensive game can be split up into two classes: infinite and finite games. An infinite game is a game that can have unbounded or infinite play length, and, in the context of a normative multi-agent system, the result of such a game can thus be a computation (an infinite sequence of states) of the system. A finite game is a game that has a definite beginning and ending, and the result of such a game is always a history (a finite sequence of states) of the system. The work in this chapter is concerned with the latter, we only consider finite games. In [Bulling and Dastani, 2011b], mechanism design in the context of infinite computations is discussed, leading to a fundamentally different framework. Because the execution of a multi-agent system can, in general, be infinite, we need an explicit notion of the *beginning* and *ending* of a game. The underlying idea is as follows:

1. We only look at *some significant* finite part of the execution, that is, the execution up until a certain pre-determined point.

2. We *assume* that the agents are *only* concerned with this finite part of the execution.

Thus, we only focus our attention to some significant finite part of the execution. In the presence of a sequence-based norm, the part of the execution we are interested in are all the histories that end when an outcome is reached. We can then assume that an agent prefers a history over another history if he prefers the outcome (positive/negative sanctions) achieved at the terminal state of this history over the outcome (positive/negative sanctions) achieved at the terminal state of the other history. This leads to a class of games which we will call the *normative games*. From these games, there is one game in particular that is most interesting to us, and that is the *minimal normative game*. The minimal normative game is the normative game 'with the shortest size', that is, the normative game which ends at *the first occurring state* where a soft state is assigned by the norm. If the set of normative games is non-empty, the minimal game always exists and is unique (which will be shown later). The underlying assumption we make with the minimal game is that the agents care only about the sanctions in the nearest future, and not about the sanctions that can be achieved afterwards. If we are interested in what happens afterwards, we can apply the same process in which we look at the minimal normative game, but now take as initial state the last occurring state of the previous game. In other words, the minimal game carries properties of uniqueness and relevancy, and is thus a suited candidate out of all the normative games to consider. To capture these ideas, we introduce the notion of a *normative game tree*.

Definition 4.9 (Normative game tree)**.** *Given a multi-agent system* $M = (\Pi, Ags, ag, Act, H_0)$ *and a sequence-based norm* f*, a finite normative game tree* $T \subseteq \mathcal{H}_M$ *is a finite non-empty set of histories that adheres to the following properties:*

1. $H_0 \in T$.
2. *If* $H_0 \dots H_n H_{n+1} \in T$ $(n \geq 0)$*, then* $H_0 \dots H_n \in T$ *and* $\forall \alpha \in Act_{H_n}$ *it holds that* $H_0 \dots H_n H_n(\alpha) \in T$.
3. *If* $h \in T$ *and there exists no* $h' \in T$ *such that* $h \sqsubset h'$*, then* $h \in dom_M(f)$*. In this case, we say that* h *is* terminal.

Given a normative game tree T*, the set of terminal histories is denoted by* $\hat{T} := \{h \in T \mid \neg \exists h' \in T : h \sqsubset h'\}$*. The set of all possible normative game trees given* M *and* f *is denoted by* $\mathcal{T}_M(f)$*, which may be empty, finite or infinite. The set of minimal game trees* $\mathcal{T}_M^{min}(f)$ *is denoted as:*

$$\mathcal{T}_M^{min}(f) := \{T \in \mathcal{T}_M(f) \mid \neg \exists T' \in \mathcal{T}_M(f) : T' \subseteq T\}$$

Every history in a normative game tree represents a point in the execution of a multi-agent system in which (1) either a choice still has to be made by an agent (if this history is non-terminal), or (2) in which the game has ended and an outcome is achieved (if this history is terminal). If a history of the multi-agent system is not part of the game tree, we know that there exists a (smaller) prefix of this history which is a terminal history. The underlying idea of such a game tree is that it represents a *partial unwinding* of a transition system to a tree that is bi-similar, a common practice that is for example done in Computational Tree Logic, see [Clarke et al., 2001]. Let us give a short textual description of each of the items in the definition:

1. The first item in the definition states that the initial state of the multi-agent system is always part of the game tree.

2. The second item in the definition defines two important properties of a given tree T, namely:

   - (*Full connectedness*) If $H_0 \dots H_n H_{n+1} \in T$ $(n \geq 0)$, then $H_0 \dots H_n \in T$. This corresponds to the fact that every (proper) prefix of a history is also in the tree. This implies that there are no 'holes in the tree', or, any history in the game tree (apart from initial history H_0) can be reached by traversing a set of non-terminal histories all included in the game tree.
   - (*Preservation of choice*) If $H_0 \dots H_n H_{n+1} \in T$ $(n \geq 0)$, then $\forall \alpha \in Act_{H_n}$ it holds that we have $H_0 \dots H_n H_n(\alpha) \in T$. This corresponds to the fact that if we are in a non-terminal history, and we perform an arbitrary action, we always end up in a history that is also in the game tree. In other words, the agents maintain the same choices in a game tree as they had in the original multi-agent system.

3. The third and final item in the definition states that every terminal history in the game tree should end in a hard state in which a soft state is assigned by the norm.

It is important to note that the set of possible normative game trees may be empty. Informally, this happens if there is a way of traversing the states in the multi-agent system such that a soft state is never encountered. To formally capture this idea, we introduce the concept of a *run* of a multi-agent system. A run is a sequence of states, starting from the initial state of the multi-agent system, which is either infinite (goes on forever), or is finite and the last occurring state has no applicable actions. Note that this definition of a run is slightly different than the definition of a run of a concurrent game structure we saw in the previous chapter (i.e. Definition 3.2), because we now

allow finite runs to occur if we arrive in a deadlock state. Formally, a *run* of a multi-agent system is defined in the following way.

Definition 4.10 (Runs)**.** *Given a multi-agent system* $M = (\Pi, Ags, ag, Act, H_0)$*, a* run r *is sequence of states starting from initial state* H_0 *which is either . . . :*

- *. . .*finite*, that is* $r = H_0 \dots H_n$*, and for which* $H_0 \dots H_n \in \mathcal{H}_M$ *and* $Act_{H_n} = \varnothing$*, or;*
- *. . .*infinite*, that is* $r = H_0 H_1 \dots$*, and for which for each position* $t \geq 0$ *it holds that* $\exists \alpha \in Act_{H_t}$ *such that* $H_t(\alpha) = H_{t+1}$*.*

Given a history $h \in \mathcal{H}_M$ *and run* $r \in \mathcal{R}_M$*, we use notation* $h \sqsubseteq r$ *to denote that* h *is a finite prefix of* r*. The set of all runs given* M *is denoted by* $\mathcal{R}_M$*.*

We can use this concept of a run to characterize when the set of normative game trees is empty, as shown in the following proposition.

Proposition 4.3. *Given a multi-agent system* M *and sequence-based norm* f*, we have* $\mathcal{T}_M(f) = \varnothing$ *iff there exists a run* $r \in \mathcal{R}_M$ *such that* $\forall h \sqsubseteq r : h \notin dom_M(f)$*.*

Proof. We show left-to-right ($\Rightarrow$) and right-to-left ($\Leftarrow$) separately:

$\Rightarrow$ Suppose the contrary, i.e. suppose that for all runs $r \in \mathcal{R}_M$ we have that there exists a $h \sqsubseteq r$ such that $h \in dom_M(f)$. Given such an r, let $h_r^{min} \sqsubseteq r$ be the history for which $h_r^{min} \in dom_M(f)$ and for which $\neg \exists h' \in dom_M(f)$ such that $h' \sqsubset h_r^{min}$. In words, h_r^{min} is the smallest finite prefix of r which is in the domain of f. We now construct the following game tree $T := \{h \mid \exists r \in \mathcal{R}_M : h \sqsubseteq h_r^{min}\}$, and claim that $T \in \mathcal{T}_M(f)$, and thus that $\mathcal{T}_M(f) \neq \varnothing$, as needed. To see why this is the case, we have to verify the three requirements of Definition 4.9. The interesting case is the second requirement, which demands that T is fully connected and that T preserves choice. We show that T preservers choice, and leave the other requirements to the reader. Suppose we have an arbitrary history $H_0 \dots H_n H_{n+1} \in T$, and suppose we take an arbitrary action $\alpha \in Act_{H_n}$. Since there does not exist a $h \sqsubseteq H_0 \dots H_n$ such that $h \in dom_M(f)$ (by construction of T), and since for a certain run $r \in \mathcal{R}_M$ we have that $H_0 \dots H_n H_n(\alpha) \sqsubseteq r$, we can conclude that $H_0 \dots H_n H_n(\alpha) \sqsubseteq h_r^{min}$, and thus that $H_0 \dots H_n H_n(\alpha) \in T$. Since this history and action were chosen arbitrarily, we know that this argument works for every history and action.

⇐ Let $r \in \mathcal{R}_M$ be the run for which we have that $\forall h \sqsubseteq r : h \notin dom_M(f)$, and assume an arbitrary set $T \subseteq \mathcal{H}_M$ which adheres to requirement 1 and 2 of Definition 4.9 (implying that $T \neq \varnothing$). We show that T can never adhere to requirement 3, and thus that $T \notin \mathcal{T}_M(f)$, allowing us to conclude that $\mathcal{T}_M(f) = \varnothing$ as needed. Particularly, we have that $\exists h \in \hat{T}$ (where $\hat{T}$ denote the terminal histories of T) such that $h \sqsubseteq r$, allowing us to immediately conclude that $h \notin dom_M(f)$.

□

Again in words, this proposition states that the set of normative game trees is empty if and only if there is a way of traversing the states in the multi-agent system such that a soft state is never encountered. A very specific instantiation of this proposition happens when we let f be the empty function (the sequence-based norm whose domain is empty), since $dom_M(f) = \varnothing$ implies that for any run r we have that $\forall h \sqsubseteq r : h \notin dom_M(f)$, regardless of the system M. Thus, in order to extract a relevant game tree from the multi-agent system which we want to analyse, we need the norm and the multi-agent system to 'fit together' in some manner, i.e. for a given multi-agent system we cannot use any arbitrary sequence-based norm. Since it holds that $\mathcal{T}_M^{min}(f) \subseteq \mathcal{T}_M(f)$, whenever $\mathcal{T}_M(f)$ is empty, $\mathcal{T}_M^{min}(f)$ is also empty. We have the following proposition for minimal game trees, which states that the minimal game tree is always *unique*.

Proposition 4.4. *Given a multi-agent system M and sequence-based norm f, whenever $\mathcal{T}_M(f) \neq \varnothing$, the set $\mathcal{T}_M^{min}(f)$ is a* singleton.

The reason why this holds is because of the underlying idea that we can continuously unravel the multi-agent system one history at a time, starting from the initial history consisting of just the initial hard state, by applying all possible actions and adding them to the game tree. Whenever we encounter a history which is in the domain of the sequence-based norm, we continue with another history, until no more histories can be added to the game tree, i.e. every terminal history in the current game tree in the domain of the sequence-based norm. We know that this will always eventually happen, since the set of all possible game trees is non-empty. Moreover, since we know that this process of unravelling is deterministic, we always end up with a unique minimal game tree.

With the definition of a normative game tree, we can define the normative games we are interested in. Formally an extensive normative game (with perfect and complete information, an assumption which will be discussed later) is defined as follows.

Definition 4.11 (Extensive normative game with perfect and complete information)**.** *A finite extensive normative game with perfect and complete information is a triple $G = (M_{\succsim}, f, T)$ for which $T \in \mathcal{T}_M(f)$. A strategy σ_i in this game for an agent $i \in Ags$ is a mapping from every non-terminal history $h \in T \backslash \hat{T}$ for which $ag(H_h) = i$ to an action $\alpha \in Act_{H_h}$, and we define the set of strategies for i in G as $\Sigma_i(G)$. A strategy profile $\sigma = (\sigma_1, \ldots, \sigma_{|Ags|})$ is a tuple of strategies for each agent, and likewise we use $\Sigma(G)$ to denote the set of all strategy profiles. A strategy profile $\sigma \in \Sigma(G)$ generates a unique terminal history $run(\sigma) = H_0 \ldots H_n \in \hat{T}$ for which it holds that $H_t(\sigma_{ag(H_t)}(H_0 \ldots H_t)) = H_{t+1}$ $(0 \leq t < n)$. Given two strategy profiles $\sigma, \sigma' \in \Sigma(G)$, we have that agent $i \in Ags$ prefers σ at least as much as σ', denoted $\sigma \succsim_i \sigma'$, if and only if $f(run(\sigma)) \succsim_i f(run(\sigma'))$. The length of a game $G = (M_{\succsim}, f, T)$, notation $|G|$, is defined as the largest possible number of actions that needs to be performed until the game ends, and is defined as follows:*

$$|(M_{\succsim}, f, T)| := \max_{h \in T}(|h|)$$

Given a multi-agent preference structure $M_{\succsim}$ and a propositional formula $\psi \in \mathcal{L}_{prop}(\Pi)$, the set of all possible games we can construct by taking an arbitrary sequence-based norm from $\mathcal{N}_M(\psi)$ is denoted by $\mathcal{G}_{M_{\succsim}}(\psi)$. This set is defined as follows:

$$\mathcal{G}_{M_{\succsim}}(\psi) := \bigcup_{f \in \mathcal{N}_M(\psi)} \{(M_{\succsim}, f, T) \mid T \in \mathcal{T}_M(f)\}$$

The minimal games $\mathcal{G}^{min}_{M_{\succsim}}(\psi)$ are likewise defined as:

$$\mathcal{G}^{min}_{M_{\succsim}}(\psi) := \bigcup_{f \in \mathcal{N}_M(\psi)} \{(M_{\succsim}, f, T) \mid T \in \mathcal{T}^{min}_M(f)\}$$

Note that we use the same notation $* \succsim_i *$ for both soft states ($S \succsim_i S'$: agent i prefers soft state S at least as much as S') and strategies ($\sigma \succsim_i \sigma'$: agent i prefers strategy profile σ at least as much as σ'), but if we are careful with our syntax this should not lead to any confusion. In general, the set $\mathcal{G}^{min}_{M_{\succsim}}(\psi)$ may be empty, be a singleton or contain multiple elements. However, we have the following proposition which states that the minimal game is uniquely defined (i.e. the set $\mathcal{G}^{min}_{M_{\succsim}}(\psi)$ is a singleton) if ψ is a state based norm and if the set of normative game trees is non-empty.

Proposition 4.5. *Given a multi-agent preference structure $M_{\succsim}$ and propositional formula $\psi \in \mathcal{L}_{prop}(\Pi)$, if it is the case that ψ is a state-based norm, that is $\mathcal{N}_M(\psi) = \{f'\}$, and $\mathcal{T}_M(f) \neq \varnothing$, then $\mathcal{G}^{min}_{M_{\succsim}}(\psi)$ is a singleton.*

Proof. From the fact that ψ is a state-based norm, we can conclude by definition that $\mathcal{G}^{min}_{M_{\succsim}}(\psi) = \{(M_{\succsim}, f', T) \mid T \in \mathcal{T}^{min}_M(f)\}$. However, due to Proposition 4.4 we have from $\mathcal{T}_M(f) \neq \varnothing$ that $\mathcal{T}^{min}_M(f)$ is a singleton, i.e. $\mathcal{T}^{min}_M(f) = \{T'\}$, and thus we have that $\mathcal{G}^{min}_{M_{\succsim}}(\psi) = \{(M_{\succsim}, f', T')\}$, as required. □

The notation $\mathcal{G}^{min}_{M_\succsim}(\psi)$ (and $\mathcal{G}_{M_\succsim}(\psi)$) will play an important role in the upcoming sections of this chapter. We make the assumption that the agents have perfect and complete information. From a game-theoretic perspective, this means the following:

- **Complete information**: A game with complete information is such that the preference of every agent and the structure of the game is *commonly known among the agents*.
- **Perfect information**: A game with perfect information is such that at any stage of the game, it is *commonly known among the agents* what has taken place earlier in the game. Thus, every agent exactly knows the actions that have occurred in the past, and they know that they know this (etcetera).

Especially the assumption of complete information may seem strong. However, it is important to note that we only assume that the agents participating in the multi-agent system have complete information. We do not assume that the designer of the system has this same information, i.e. we may assume that the designer is unsure what the true preferences of the agents are. Later when we reach the topic of mechanism design, we will see that we can model this uncertainty by considering a set of possible preference profiles of the agents.

In order to predict the possible ways a game will be played out, we can use game-theoretic solution concepts to decide the possible strategies the agents will adopt, and therefore the result of the game. The solution concepts we use in this chapter are that of Nash Equilibrium (NE) and Sub-game Perfect Nash Equilibrium (SPNE). A detailed introduction of these concepts can be found in [Osborne and Rubinstein, 1994]. A strategy profile constitutes a Nash Equilibrium in a game if no agent has anything to gain by changing his or her own strategy. If this is the case, we can simply say that this strategy profile is a Nash Equilibrium strategy profile in this game. A strategy profile constitutes a Sub-game Perfect Nash Equilibrium if this strategy profile constitutes a Nash Equilibrium in every sub-game of the original game. A sub-game of an extensive game is any possible smaller game that arises after the performance of zero or more actions from the agent. Again, if this is the case, we call such a strategy profile a Sub-game Perfect Nash Equilibrium strategy profile.

Definition 4.12 ((Sub-game Perfect) Nash Equilibrium)**.** *Given a game $G = (M_\succsim, f, T)$, strategy profile $\sigma \in \Sigma(G)$ and strategy $\sigma_i \in \Sigma_i(G)$ for agent i, let $\sigma[\sigma_i] \in \Sigma(G)$ be equal to σ except where the strategy of agent i is replaced by σ_i. The set of Nash Equilibrium (NE) strategy profiles given a game G, denoted $\Sigma_{NE}(G) \subseteq \Sigma(G)$, is defined as:*

$$\Sigma_{NE}(G) := \{\sigma \in \Sigma(G) \mid \forall i \in Ags, \sigma_i \in \Sigma_i(G) : \sigma \succsim_i \sigma[\sigma_i]\}$$

Given a game $G = (M_{\succsim}, f, T)$ *and history* $h \in T$*, we let* $G_h = ((M, H_h)_{\succsim}, f, \{h' \mid (h + h') \in T\})$ *denote the sub-game of* G *starting from history* h*. The set of Sub-game Perfect Nash Equilibrium (SPNE) strategy profiles given a game* G*, denoted* $\Sigma_{SPNE}(G) \subseteq \Sigma_{NE}(G) \subseteq \Sigma(G)$*, is defined as:*

$$\Sigma_{SPNE}(G) \coloneqq \{\sigma = (\sigma_1, \ldots, \sigma_{|Ags|}) \in \Sigma(G) \mid \forall h \in T : \sigma_h \in \Sigma_{NE}(G_h)\},$$
$$\textit{where } \sigma_h = (\sigma'_1, \ldots, \sigma'_{|Ags|}) \textit{ such that } \sigma'_i(*) = \sigma_i(h + *)$$

A known game-theoretic result is that for every extensive-form game a NE and SPNE strategy profile always exists. Since our extensive normative games can be specified as such, this result transfers. That is, given a game G, we have that $\Sigma_{\text{NE}}(G) \neq \varnothing$ and $\Sigma_{\text{SPNE}}(G) \neq \varnothing$.

In this section we discussed how a normative multi-agent system leads to a (set of) normative game(s). In this thesis, we are interested in formal verification of these systems. In the next section, we will frame our verification problem using principles from *mechanism design*, which we have already discussed in the introduction of this chapter.

4.3 Mechanism design

In this section, we will frame our verification problem using the principles from mechanism design and implementation theory. For a more general introduction to mechanism design, we direct the reader to the introduction of this chapter. We will now briefly repeat some of the key points that were discussed in the introduction. Readers that are already familiar with the key concepts of mechanism design may skip to the next paragraph. The question we want to answer is given a goal, can we design and verify a mechanism whose *predicted* outcomes coincide with the *desirable* outcomes, that is, the outcomes which are aligned with our goal? We will now give a short recap of the introduction to mechanism design we gave in the beginning of this chapter. The outcomes we consider in this chapter are the soft facts (the positive and negative sanctions) imposed on the agents. Moreover, the mechanisms we use are the state-based norms, which tell the agents which states (situations) should be achieved or avoided by associating to these states certain sanctions. The predicted outcomes are the outcomes we predict will occur, and we can derive these by using concepts from game theory. In this work, we concern ourselves with the *minimal normative games*. As mentioned in the previous section, the basic underlying assumption we make with these kinds of games is that the agents only care about the sanctions in the nearest future, and not about the sanctions that can be achieved at any other point in the future. The knowledge assumptions we make of the agents are that they have *complete* and *perfect* information of the games they are playing, and the solution

concepts we consider are that of *sub-game perfect Nash equilibria*. The desired outcomes are the outcomes we want to occur. In general, the desired outcomes are the outcomes that maintain order in society. Because mechanism designers do not know the agent types beforehand, a more cautious approach has to be employed. We do not know beforehand if an agent has the incentive of committing a crime, and as such, we cannot simply prescribe sanctions before any execution has occurred. This information must slowly be generated as the system is executed. We can now write it down formally.

To model the various agent types, we assume we have a set Θ of these types. Since we assume in this work that all the agents have complete and perfect information, we can view a type simply as a preference profile (a preference for each agent). Thus, the set of types Θ consists of all possible preference profiles the designer of the system considers possible. As mentioned earlier, we do not wish to focus our attention on formulating criteria specifying the optimal outcomes. As such, we may assume we have a social choice rule which maps possible agent types to a formula representing the possible outcomes which would be optimal given this type.

Definition 4.13 (Social choice)**.** *Given a multi-agent system M and a non-empty set Θ of possible preference profiles of the agents. A social choice rule $sc : \Theta \to \mathcal{L}_{prop}(\Pi_{\text{soft}})$ associates with each possible type $\succsim \in \Theta$ a satisfiable logical formula over soft facts $sc(\succsim)$ representing a (non-empty) set of social optimal soft states. We use the notation $\mathcal{O}_{sc(\succsim)}$ to refer these soft states, i.e.:*

$$\mathcal{O}_{sc(\succsim)} = \{S \in \mathcal{P}(\Pi_{\text{soft}}) \mid S \vDash_{prop} sc(\succsim)\}$$

Given a game G, the achieved outcomes by executing a sub-game perfect Nash Equilibrium strategy profile, denoted $\mathcal{O}_{SPNE}(G)$, is defined as:

$$\mathcal{O}_{SPNE}(G) := \{f(run(\sigma)) \mid \sigma \in \Sigma_{SPNE}(G)\}$$

A type $\succsim \in \Theta$, a state-based norm ψ and a multi-agent system M lead to the set of achieved outcomes by the minimal game. Remember however that the minimal game may not exist if the set of normative game trees is empty, which is characterized by Proposition 4.3. However, if there exists a $G \in \mathcal{G}^{min}_{M_\succsim}(\psi)$, we can derive $\mathcal{O}_{\text{SPNE}}(G)$. The question now becomes whether, given a type $\succsim \in \Theta$, the *desired outcomes* $\mathcal{O}_{sc(\succsim)}$ match the *achieved SPNE outcomes* $\mathcal{O}_{\text{SPNE}}(G)$. This procedure is drawn in Figure 4.1. The idea of normative mechanism design was first coined in [Bulling and Dastani, 2011a], and our idea relates to this idea in the sense that we are also looking at environment programs extended with norms that implement a social choice function (in their work "normative behaviour function") in equilibrium.

In general, one might look at different implementability criteria varying in strength. In *partial implementation*, given a norm, we want to verify whether

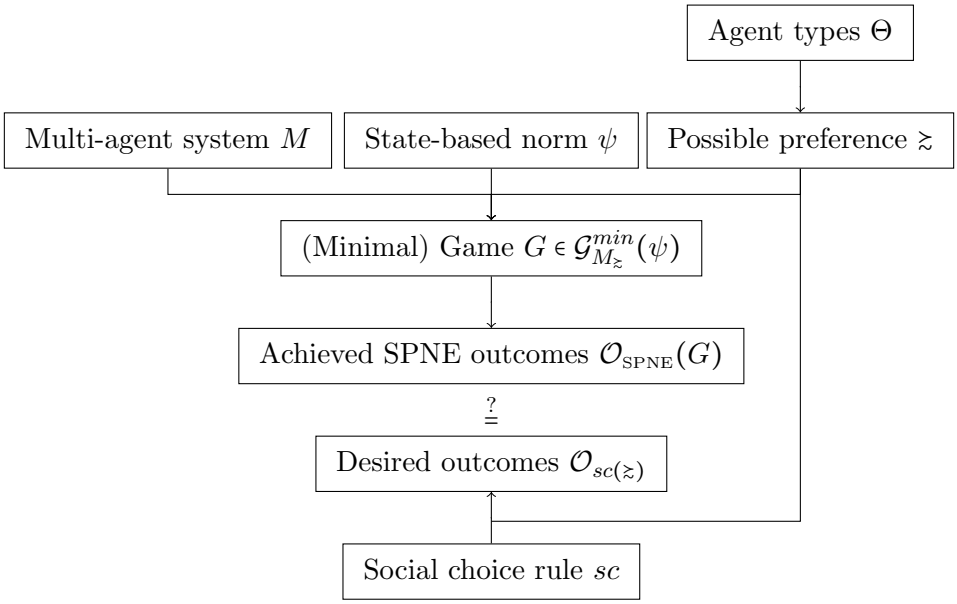


Figure 4.1: Figure giving a basic overview of the implementation framework we adopt in this chapter. The direction of the arrows should be read as "gives rise to".

all the possible social optimal outcomes are achieved by some SPNE strategy profile. The underlying idea is that all the social optimal outcomes are equally optimal for society, and we want all of them to be realized by at least one equilibrium strategy profile. In *weak implementation*, we want to verify whether all SPNE strategies lead to a social optimal outcome. The underlying idea here is that all outcomes produced by a SPNE strategy profile can occur, and we want all of them to be socially optimal. There is one additional difference between partial and weak implementation in this framework. In partial implementation, we demand that the minimal game always exists, i.e. that the set of minimal normative game trees is non empty. In weak implementation, we do not set this restriction, that is, we also say that a norm weakly implements a social choice rule even though the minimal game does not exist. However, in *full implementation*, we demand both these criteria. That is, we demand that the set of social optimal outcomes *coincides* with the set of SPNE outcomes, and that the minimal game exist.

Definition 4.14 (Implementability)**.** *Given a multi-agent system M and a set of agent types Θ, we say that a norm $\psi \in \mathcal{L}_{prop}(\Pi)$ ...:*

1. ***Partial:*** *... partially SPNE-implements sc in M iff for all $\succsim \in \Theta$ we*

have that $\exists G \in \mathcal{G}^{min}_{M_{\gtrsim}}(\psi)$ *such that:*

$$\mathcal{O}_{sc(\gtrsim)} \subseteq \mathcal{O}_{SPNE}(G)$$

2. ***Weak:*** *... weakly SPNE-implements sc in M iff for all* $\gtrsim \in \Theta$ *we have that* $\forall G \in \mathcal{G}^{min}_{M_{\gtrsim}}(\psi)$*:*

$$\mathcal{O}_{SPNE}(G) \subseteq \mathcal{O}_{sc(\gtrsim)}$$

3. ***Full:*** *... (fully) SPNE-implements sc in M iff* ψ *both partially and weakly SPNE-implements sc in M. Alternatively, for all* $\gtrsim \in \Theta$ *we have that* $\exists G \in \mathcal{G}^{min}_{M_{\gtrsim}}(\psi)$ *such that:*

$$\mathcal{O}_{SPNE}(G) = \mathcal{O}_{sc(\gtrsim)}$$

Observe that the logical definition of partial and weak implementation indeed imply the logical definition of full implementation. The reason for this is because for a given multi-agent preference structure $M_{\gtrsim}$ and state-based norm ψ, the set $\mathcal{G}^{min}_{M_{\gtrsim}}(\psi)$ is either empty or a singleton. We will use the famous example of Solomon's Dilemma as an example to illustrate the ideas presented in this chapter up until this point. At the end of this chapter after developing our calculus, we will come back to the dilemma.

4.3.1 Example of Solomon's dilemma

Solomon's dilemma is often used in literature to describe the idea of implementation theory and mechanism design, see for example [Osborne and Rubinstein, 1994]. In this dilemma two women come before him, both claiming to be the mother of a child, and Solomon has to allocate the child to one of the mothers. In this chapter we consider the problem where Solomon is able to give the mothers a fine, as is discussed in [Moore, 1992], since the version in which Solomon is able to cut the child in half (the biblical version) has no sub-game perfect solution. It is clear that if mother 1 is the true mother, then the optimal outcome would be that mother 1 gets the child, and that both mothers never receive any fine. If mother 2 is the true mother, then the optimal outcome would be that mother 2 eventually gets the child, and again that both mothers never receive any fine. This example makes it clear why a social choice rule is dependent on the preference profile: since we do not know who the real mother is, we cannot simply say that there exists one unique optimal outcome. Each mother (starting with mother 1 in hard state $\varnothing$) takes turns in either confirming whether she is the mother or denying this. The multi-agent system (Π, Ags, ag, Act, H_0) representing the scenario is defined as follows:

- $\Pi = \Pi_{\text{hard}} \cup \Pi_{\text{soft}}$. We have $\Pi_{\text{hard}} = \{s_1, s_2, m_1, m_2\}$, where the atomic proposition s_i ($i = 1, 2$) stands for whether mother i is the last one who has spoken and atomic proposition m_i stands for whether mother i in her last claim confirmed that she is the real mother. Moreover, $\Pi_{\text{soft}} = \{o_{\text{child}_1}, o_{\text{fine}_1}, o_{\text{fine}_2}\}$, where o_{child_1} denotes the soft fact in which mother 1 gets the child (true meaning mother 1, false meaning mother 2), o_{fine_1} the soft fact representing a small fine that can be given to mother 1, and o_{fine_2} the soft fact representing a big fine that can be given to mother 2.
- $Ags = \{1, 2\}$
- $$ag(H) = \begin{cases} 1 & \text{if } s_1 \notin H \\ 2 & \text{otherwise} \end{cases}$$
  That is, if mother 1 has not just spoken, it is her turn, otherwise mother 2 gets to speak.
- $$Act = \{(\neg s_1, \{s_1, m_1\}, \{s_2\}), (s_1, \{s_2, m_2\}, \{s_1\}), (\neg s_1, \{s_1\}, \{s_2, m_1\}), (s_1, \{s_2\}, \{s_1, m_2\})\}$$
  The actions with precondition $\neg s_1$ represent the actions of mother 1, which can either claim to be the real mother (make m_1 true) or not (make m_1) false. Likewise, the actions with precondition s_1 represent the actions of mother 2, which can either make m_2 true or false.
- $H_0 = \varnothing$ is the initial hard state where both of the mothers have not spoken and have not made a claim about being the real mother yet.

As we will see later, o_{fine_2} is tweaked precisely by Solomon such that this fine is low enough that if mother 2 would be the real mother, she would care more about the child, while if mother 2 would *not* be the real mother, she would care more about the sanction. Of course this requires some accurate and knowledgeable estimations by Solomon, but we assume that he is wise enough to do this. The role of o_{fine_1} is to disincentivize mother 1 to claim she is the real mother when she is in fact not. The complete multi-agent system is drawn in Figure 4.2, where the agent who decides on an action is shown above each state. Each state has two outgoing actions, particularly one where each mother can confirm that she is the real mother (make m_i true and s_i true), and one where she denies this (make m_i false and s_i true). As a convention, whenever we draw a multi-agent system, we only draw the states that are reachable from initial state $\varnothing$, which is why for example state $\{s_1, s_2, m_1, m_2\}$

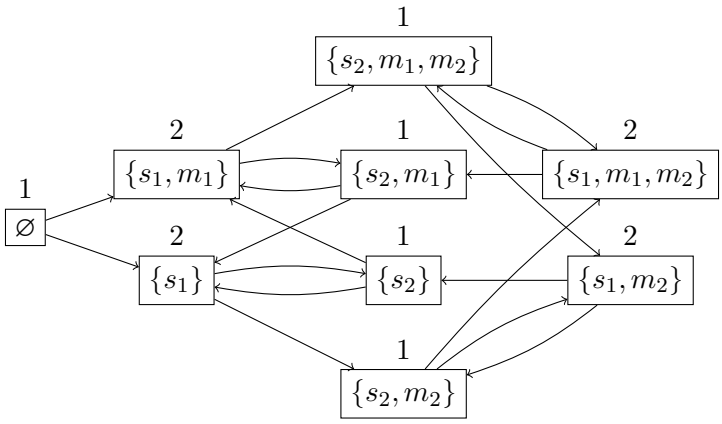


Figure 4.2: The multi-agent system describing Solomon's dilemma. Each mother takes turns in either confirming whether she is the mother or denying this.

is not drawn. The king considers the following two true preference profiles $\Theta = \{\succsim, \succsim'\}$:

1. The preference profile in which mother 1 is the real mother:

   $\succsim = (\succsim_1, \succsim_2)$, such that $\{o_{\text{child}_1}\} \succ_1 \{o_{\text{child}_1}, o_{\text{fine}_1}\} \succ_1 \varnothing \succ_1 \{o_{\text{fine}_1}\}$, and,
   $\varnothing \succ_2 \{o_{\text{child}_1}\} \succ_2 \{o_{\text{fine}_2}\} \succ_2 \{o_{\text{child}_1}, o_{\text{fine}_2}\}$

   Mother 1 always prefers getting her child over the small fine the king can give to her and mother 2 prefers that the child is given to mother 1 over getting the child and receiving a big fine ($\{o_{\text{child}_1}\} \succ_2 \{o_{\text{fine}_2}\}$). Note that for example for soft state $\varnothing$ the soft fact o_{child_1} is false, and thus this represents a soft state in which the child is given to mother 2. Moreover, we assume mother 1 is indifferent about o_{fine_2} since she does not care about the fine that mother 2 gets (formally, for every soft state $S \in \mathcal{P}(\Pi_{\text{soft}})$ we have $S \sim_1 (S \cup \{o_{\text{fine}_2}\})$), and likewise for mother 2 about o_{fine_1} (for every $S \in \mathcal{P}(\Pi_{\text{soft}})$ we have $S \sim_2 (S \cup \{o_{\text{fine}_1}\})$). By using these two rules and by using the assumption that the preference relations are reflexive, each of the above stated preferences can be extended to be a *complete*, *transitive* and *reflexive* relation over all soft states, as required by Definition 4.8.

2. The preference profile in which mother 2 is the real mother:

   $\succsim' = (\succsim'_1, \succsim'_2)$, such that $\{o_{\text{child}_1}\} \succ'_1 \{o_{\text{child}_1}, o_{\text{fine}_1}\} \succ'_1 \varnothing \succ'_1 \{o_{\text{fine}_1}\}$, and,
   $\varnothing \succ'_2 \{o_{\text{fine}_2}\} \succ'_2 \{o_{\text{child}_1}\} \succ'_2 \{o_{\text{child}_1}, o_{\text{fine}_2}\}$

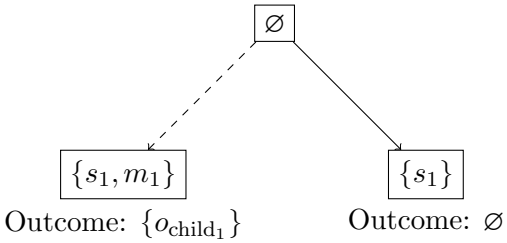


Figure 4.3: The game $G \in \mathcal{G}^{min}_{M_{\gtrsim}}(\psi)$ as explained in the text, where we have that $\{o_{\text{child}_1}\} \succ_1 \varnothing$. The dashed line represents the optimal action for mother 1, namely claiming to be the real mother.

> The only difference in this preference profile is the fact that now $\{o_{\text{fine}_2}\} \succ'_2 \{o_{\text{child}_1}\}$, since in this case mother 2 does not care about the big fine, she would rather have her child. We still assume that these preference relations are reflexive, and that each mother is indifferent about the fine of the other mother.

The first preference $\gtrsim_2$ states that mother 2 prefers an outcome over any other outcome if she is assigned the child without a fine given. If this is not the case, she would rather not receive a fine. The second preference $\gtrsim'_2$ states that mother 2 prefers an outcome in which she is assigned the child, regardless of whether this outcome contains a fine or not, while the rest remains the same. In other words, he either considers that mother 2 cares more about the fine than the child, or more about the child than the fine; the first case represents the case in which mother 1 is the real mother, while the second case represents the case in which mother 2 is the real mother. Solomon's goal is to implement the following social choice rule *sc*:

1. $sc(\gtrsim) = o_{\text{child}_1} \wedge \neg o_{\text{fine}_1} \wedge \neg o_{\text{fine}_2}$. That is, if mother 1 is the real mother then she should get the child without any fines given to either mother. We have $\mathcal{O}_{sc(\gtrsim)} = \{o_{\text{child}_1}\}$.

2. $sc(\gtrsim') = \neg o_{\text{child}_1} \wedge \neg o_{\text{fine}_1} \wedge \neg o_{\text{fine}_2}$. That is, if mother 2 is the real mother then she should get the child without any fines given to either mother. We have $\mathcal{O}_{sc(\gtrsim)} = \varnothing$.

The question we would like to answer is whether Solomon can devise a norm that SPNE implements the social choice rule *sc*; i.e. devise a norm such that if the mothers act rationally (according to a subgame perfect equilibrium), the child is always giving to the correct mother and no fines are given. A naive attempt by Solomon might be to use the following state-based norm ψ (where

notation ϕ^{hard}_H for a given hard state H is explained on page 64):

$$\psi = (\phi^{\text{hard}}_{\{s_1,m_1\}} \wedge sc(\succsim)) \vee (\phi^{\text{hard}}_{\{s_1\}} \wedge sc(\succsim'))$$

This norm states that any history ending in hard state $\{s_1, m_1\}$ ought to be assigned soft state $sc(\succsim)$, and any history ending in hard state $\{s_1\}$ ought to be assigned soft state $sc(\succsim')$. The set $dom_M(\psi)$ thus consists of an infinite number of histories, namely all the histories ending in hard state $\{s_1, m_1\}$ or hard state $\{s_1\}$. From Figure 4.2 we can easily see that this for example includes history $\varnothing\{s_1\}$ of length 1, but also history $\varnothing\{s_1\}\{s_2\}\{s_1\}$ of length 3. However, the unique minimal game $\mathcal{G}^{min}_{M_\succsim}(\psi) = \{(M_\succsim, f, T)\}$, which we have also drawn in Figure 4.3, looks as follows:

- $T = \{\varnothing, \varnothing\{s_1, m_1\}, \varnothing\{s_1\}\}$, i.e. starting from initial hard state $\varnothing$, mother 1 can perform one action, after which the game ends. Observe that the sequence $\varnothing\{s_1, m_1\}$ represents the history starting from initial state $\varnothing$ and ending in state $\{s_1, m_1\}$, and history $\varnothing\{s_1\}$ the history starting from initial state $\varnothing$ and ending in $\{s_1\}$.
- $\{\varnothing\{s_1, m_1\}, \varnothing\{s_1\}\} \subseteq dom_M(f)$, such that:

  $$f(\varnothing\{s_1, m_1\}) = \mathcal{O}_{sc(\succsim)} = \{o_{\text{child}_1}\}$$

  And:

  $$f(\varnothing\{s_1\}) = \mathcal{O}_{sc(\succsim')} = \varnothing$$

  That is, this state-based norm assigns mother 1 the child if she claims to be the real mother, and otherwise assigns mother 2 the child.

Note that the size of this game $|G|$ is equal to 1, since the size of the largest history in T is equal to 1. The problem with this norm is that mother 1 can always just *lie* about being the real mother, causing her to wrongfully receive the child. Since mother 2 has no opportunity to act, it is also not fair for mother 2. That is, the SPNE strategies for both games $G = (M_\succsim, f, T)$ and $G' = (M_{\succsim'}, f, T)$ coincide $\Sigma_{\text{SPNE}}(G) = \Sigma_{\text{SPNE}}(G')$, namely in both games it is optimal for mother 1 to claim to be the real mother. Thus for this particular norm ψ we have:

$$\mathcal{O}_{\text{SPNE}}(G) = \mathcal{O}_{\text{SPNE}}(G') = \mathcal{O}_{sc(\succsim)}$$
$$\mathcal{O}_{\text{SPNE}}(G') \neq \mathcal{O}_{sc(\succsim')}$$

This is why the problem of implementability often comes down to the design of a "truth-revealing" mechanism, i.e. a norm in which the agents can do no better than to reveal their true preferences. This example highlights two major difficulties that we need to tackle:

1. Given a norm, how do we (automatically) *verify* whether or not this norm implements a social choice rule in sub-game perfect equilibrium?

2. How do we *synthesize* a norm that implements a social choice rule in sub-game perfect equilibrium?

In the remainder of this chapter, we will focus our attention on the first question, and do not concern ourselves with the second question. In the next section, we will introduce a Unity-style proof system that allows us to infer what the sub-game perfect outcomes of games are. This proof system can be used to verify whether a given norm is suited to solve Solomon's Dilemma, which we will do at the end of the chapter.

4.4 Unity-style proof system for Games

In this section, we introduce a Unity-style proof system that allows us to infer what the sub-game perfect outcomes of games are. Unity, as introduced in [Chandy and Misra, 1989], is a programming language to express the execution of parallel programs in which there is no central notion of control. An accompanying proof system, in the form of a program logic was also supplied, which can be used to reason about certain invariance and reachability properties of a parallel program. These properties form the basis of the proof system we are going to use here, except we relate them to *games*. More specifically, the work in this chapter fundamentally differs from Unity on the following aspects:

- In Unity, assignments change the state-valuations, while in our setting actions change the valuations of propositions. Moreover, these actions are performed by agents.

- In Unity, there is no central notion of control. The only assumption they make in their underlying proof system is that the computations are *fair*. For example, strong fairness states that if an assignment is infinitely often applicable, then it should be applied infinitely often. In our setting, we assume that there exists agents that control the actions, i.e., the agents get to decide how the system will progress. As of such, we make no fairness assumption.

- In the proof system of Unity, the assertions that constitute this proof system generally quantify over hard states of the system. For example, a typical Unity assertion might state that for all hard states that satisfy φ, it is always the case that after an assignment has occurred, we end up in a hard state that satisfies ψ, i.e. all assignments *progress* the system

from a hard φ-state to a hard ψ-state. In our setting, the assertions will not only quantify over hard states, but also over *soft states*. The exact interpretation that needs to be given to such assertions will be discussed in the following section.

In the following section, we introduce the notion of *game triples*, which will be the basic building blocks (the axioms) of our proof system which we will develop later on in this chapter.

4.4.1 Game triples

The basic building blocks of our proof system consist of assertions of the form $\{\varphi\}M_{\gtrsim}\{\psi\}$, where $M_{\gtrsim}$ is a multi-agent preference structure, and $\varphi, \psi \in \mathcal{L}_{\text{prop}}(\Pi)$ are propositional formulas ranging over *both hard and soft facts*. The idea of using propositional formulas ranging over different types of atomic propositions (hard and soft), and using that idea to prove properties of games, can be traced back to [Pauly, 2002]. The classical interpretation of such a triple, for example in Hoare or Unity logic, would be that once we reach a state at which φ holds, then it is necessarily the case that in the next state (i.e. after every possible execution of an action) ψ holds. We attach a slightly different meaning to these triples allowing us to prove properties of games, which as we will see later *generalizes* the classical interpretation of a triple. Before we give the exact definition of when a triple $\{\varphi\}M_{\gtrsim}\{\psi\}$ is satisfied, the intuitive reading we give to it is the following: *"For any hard-soft state pair satisfying φ, the application of an optimal action will result in a hard-soft state pair satisfying ψ"*. However, this intuitive reading does not explain what we mean with an optimal action, which we will formally specify momentarily. Since the interpretation and use of these triples is different than the classical interpretation and use of these triples, we refer to these triples as *game triples*. They are defined as follows:

Definition 4.15 (Game triples)**.** *Given a multi-agent system $M = (\Pi, Ags, ag, Act, H_0)$ and a preference profile $\gtrsim$ over soft states $\mathcal{P}(\Pi_{\text{soft}})$, game triples are assertions of the form $\{\varphi\}M_{\gtrsim}\{\psi\}$, where $\varphi, \psi \in \mathcal{L}_{prop}(\Pi)$, and are defined as follows.*

$$\begin{aligned}\{\varphi\}M_{\gtrsim}\{\psi\} \;:=\; & \forall H, S \text{ for which } H \cup S \vDash_{prop} \varphi : \\ & 1.\ (\exists \alpha \in Act_H : H(\alpha) \cup S \vDash_{prop} \psi),\ \text{and,} \\ & 2.\ (\forall \alpha \in Act_H, \exists S' : H(\alpha) \cup S' \vDash_{prop} \psi \text{ and } S \gtrsim_{ag(H)} S')\end{aligned}$$

In words, the assertion $\{\varphi\}M_{\gtrsim}\{\psi\}$ states that for every hard state H and soft state S which satisfies φ, it holds that for all possible performances of actions in the next state there exists a soft state S' which satisfy ψ, but the

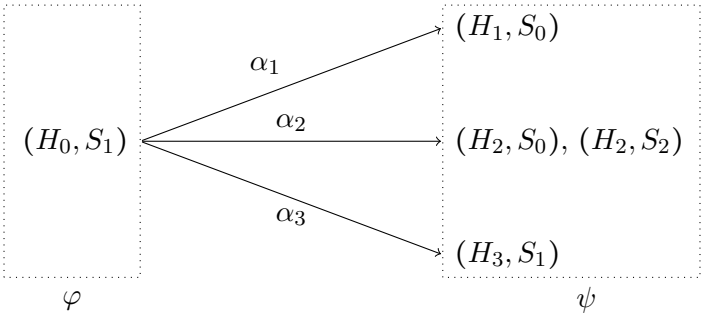


Figure 4.4: This figure illustrates the scenario we consider in Example 4.2. The box on the left show all the hard-soft state pairs satisfying φ, and the box on the right shows all the hard-soft state pairs satisfying ψ. The pairs are grouped together if they share the same hard state. The arrows show how the applicable actions change the hard states.

agent prefers S at least as much as S', and if we consider all these actions, there exists at least one action which realizes this soft state S. Note that this implies that S is not necessarily the most preferred soft state for this agent, it just implies that by *some* assignment of soft states to hard states (i.e. by some possible enforcement norm from the set $\mathcal{N}_M(\psi)$) it is the case that S is most preferred. It is important to remember that for each action we are only asking for the existence of such S', and that it does not have to hold for all soft states. In the upcoming Example 4.2 we will elaborate on this further. Additionally note that we lay no syntactic or semantic restrictions on the formulas φ and ψ, we merely assume that they are from the set $\mathcal{L}_{\text{prop}}(\Pi)$. It is very important to differentiate between the roles of the precondition φ and postcondition ψ in a game triple $\{\varphi\}M_{\tilde{\succsim}}\{\psi\}$: The precondition tells us for a hard state the soft state(s) for which we *want* to verify if they will occur, i.e. they serve a *prescriptive role*, and the postcondition tells us for a hard state the soft state that *is* realized by some assignment from hard states to soft states, i.e. they serve a *descriptive role*. Below we consider an example, which tries to work out step by step how the validity of a game triple can be shown by using Definition 4.15.

Example 4.2. *Given a multi-agent system $M = (\Pi, Ags, ag, Act, H_0)$, and two formulas $\varphi, \psi \in \mathcal{L}_{prop}(\Pi)$. We assume that the set of all hard-soft state pairs satisfying φ is equal to $\{(H_0, S_1)\}$, i.e. $H_0 \cup S_1 \vDash_{prop} \varphi$ and no other hard-soft state pairs satisfy φ, and assume that the set of all hard-soft state pairs satisfying ψ is equal to $\{(H_1, S_0), (H_2, S_0), (H_2, S_2), (H_3, S_1)\}$. Moreover, we assume there is a single agent $Ags = \{1\}$, and we assume that this agent has*

three applicable actions in H_0, particularly $Act_{H_0} = \{\alpha_1, \alpha_2, \alpha_3\}$, for which we have that $H_0(\alpha_1) = H_1$, $H_0(\alpha_2) = H_2$, and $H_0(\alpha_3) = H_3$. This situation is depicted in Figure 4.4, where the hard-soft state pairs are grouped together if they share the same hard state, and where the arrows show how the applicable actions change the hard states. We consider the following two preference relations over S_0, S_1 and S_2 for agent 1:

- $\gtrsim = (\gtrsim_1)$, *such that* $S_2 \succ_1 S_1 \succ_1 S_0$.
- $\gtrsim' = (\gtrsim'_1)$, *such that* $S_0 \succ'_1 S_1 \succ'_1 S_2$.

We ask ourselves whether $\{\varphi\}M_{\gtrsim}\{\psi\}$ and/or whether $\{\varphi\}M_{\gtrsim'}\{\psi\}$ is the case. This example uses Definition 4.15 to try and show which of these two game triples are valid, and we consider them separately below:

- *Is it the case that $\{\varphi\}M_{\gtrsim}\{\psi\}$? The first criterion of Definition 4.15 is met, namely we have that $H_0(\alpha_3) = H_3$ and $H_3 \cup S_1 \vDash \psi$. In order to show the second criterion, we consider each action separately:*

  1. *For action α_1 we have that $H_0(\alpha_1) = H_1$, and we have that there exists a soft state, namely S_0, such that $H_1 \cup S_0 \vDash \psi$ and $S_1 \gtrsim_1 S_0$. In other words, α_1 satisfies the criterion.*
  2. *For action α_2 we have that $H_0(\alpha_2) = H_2$, and we have that there exists a soft state, namely S_0, such that $H_2 \cup S_0 \vDash \psi$ and $S_1 \gtrsim_1 S_0$. Note, and this is of crucial importance, that if we would have considered S_2, we have that $H_2 \cup S_2 \vDash \psi$, but also $S_2 \not\gtrsim_1 S_0$ would be the case, and thus this soft state would not have been a good candidate to consider. This also becomes apparent when looking at Definition 4.15, particularly we are only looking for the* existence *of a soft state. In other words, α_2 satisfies the criterion.*
  3. *For action α_3 we have that $H_0(\alpha_3) = H_3$, and we have that there exists a soft state, namely S_1, such that $H_3 \cup S_1 \vDash \psi$ and $S_1 \gtrsim_1 S_1$. The latter is valid because a preference relation is always reflexive, i.e. $S_1 \sim S_1$. In other words, α_3 satisfies the criterion.*

  This is what we needed to show in order to conclude that $\{\varphi\}M_{\gtrsim}\{\psi\}$.

- *Is it the case that $\{\varphi\}M_{\gtrsim'}\{\psi\}$? The first criterion of Definition 4.15 is again met, namely we have that $H_0(\alpha_3) = H_3$ and $H_3 \cup S_1 \vDash \psi$. However, the second criterion is not met, and to see why, we consider action α_1. There is only one candidate soft state to consider, namely S_0, for which we have $H_0(\alpha_1) \cup S_0 \vDash \psi$. However, for this particular soft state, we have that $S_1 \not\gtrsim'_1 S_0$. In other words, for all soft states S', we either have that $H_0(\alpha_1) \not\vDash S'$, or we have that $S_1 \not\gtrsim'_1 S'$. In conclusion, the game triple is not valid.*

This example makes it clear that the postcondition of a game-triple need *not* be a functional formula. Let us return to the multi-agent system of Solomon's Dilemma, and see how we can use game triples to show properties of this system.

Example 4.3. *We return to the example multi-agent system of Solomon's Dilemma, introduced in Section 4.3.1 (p 75). We show that the following assertion is valid (where notation ϕ_H^{hard} and ϕ_S^{soft} is explained on page 64):*

$$\{\phi^{hard}_{\{s_2\}} \wedge \phi^{soft}_{\{o_{\text{child}_1}\}}\} M_{\gtrsim} \{(\phi^{hard}_{\{s_1,m_1\}} \wedge \phi^{soft}_{\{o_{\text{child}_1}\}}) \vee (\phi^{hard}_{\{s_1\}} \wedge \phi^{soft}_{\{o_{\text{fine}_1}\}}\}$$

For hard state $\{s_2\}$ (mother 2 has just spoken) and soft state $\{o_{\text{child}_1}\}$ (mother 1 is assigned the child), it necessarily holds that in the next state, which can be either $\{s_1\}$ (mother 1 has just spoken and does not claim to be the real mother), or $\{s_1, m_1\}$ (mother 1 has just spoken and claims to be the real mother), there exists a soft state which satisfies the right side of the game triple, which in the case of hard state $\{s_1, m_1\}$ is soft state $\{o_{\text{child}_1}\}$ and in the case of hard state $\{s_1\}$ is soft state $\{o_{\text{fine}_1}\}$. Moreover, it holds that the soft state $\{o_{\text{child}_1}\}$ out of these two is indeed the most preferred one for mother 1, i.e. $\{o_{\text{child}_1}\} \succ_1 \{o_{\text{fine}_1}\}$.

The following lemma shows why we call these triples *game triples*. In particular, the proposition shows that a triple encodes information about the existence of certain games of length 1, i.e. games that end after the performance of a single action. Later on in the chapter, this lemma will be used to prove several soundness and completeness properties of our proof system.

Lemma 4.1. *$\{\varphi\} M_{\gtrsim} \{\psi\}$ if and only if for every hard state H and soft state S for which $H \cup S \models_{prop} \varphi$ there exists a game $G \in \mathcal{G}_{(M,H)_{\gtrsim}}(\psi)$ such that $S \in \mathcal{O}_{SPNE}(G)$ and $|G| = 1$.*

Proof. We show the direction from left to right ($\Rightarrow$) and from right to left ($\Leftarrow$) separately.

- $\Rightarrow$ Assume $\{\varphi\} M_{\gtrsim} \{\psi\}$, and assume an arbitrary hard state H and soft state S for which $H \cup S \models_{\text{prop}} \varphi$. We construct a game $G \in \mathcal{G}_{(M,H)_{\gtrsim}}(\psi)$ such that $S \in \mathcal{O}_{\text{SPNE}}(G)$ and the game ends after the performance of one action, i.e. $|G| = 1$. From the definition of $\{\varphi\} M_{\gtrsim} \{\psi\}$ we have that (1) $\exists \alpha \in Act_H$ such that $H(\alpha) \cup S \models_{\text{prop}} \psi$, which we will refer to as action α_{optimal}, and (2) $\forall \alpha \in Act_H, \exists S'$ such that $H(\alpha) \cup S' \models_{\text{prop}} \psi$ and $S \gtrsim_{ag(H)} S'$, which we will refer to as soft state S_α. Define game $G = ((M,H)_{\gtrsim}, f, T)$, such that $T = \{H\} \cup \{HH(\alpha) \mid \alpha \in Act_H\}$, and $f(HH(\alpha))$ (for every $\alpha \in Act_H$):

$$f(HH(\alpha)) = \begin{cases} S & \text{if } \alpha = \alpha_{\text{optimal}} \\ S_\alpha & \text{otherwise} \end{cases}$$

Observe that for a given $\alpha \in Act_H$, the history $HH(\alpha)$ is the history starting in hard state H and ending in hard state $H(\alpha)$. It is clear that $f \in \mathcal{N}_M(\psi)$ and $|G| = 1$, and since for all actions $\alpha \in Act_H$ it holds that $f(HH(\alpha_{\text{optimal}})) \gtrsim_{ag(H)} f(HH(\alpha))$, we may conclude that α_{optimal} is an optimal action for agent $ag(H)$. Since $f(HH(\alpha_{\text{optimal}})) = S$, we can conclude that $S \in \mathcal{O}_{\text{SPNE}}(G)$. Since hard state H and soft state S were chosen arbitrarily, we may conclude that the above construction works for every hard state H and soft state S for which $H \cup S \models_{\text{prop}} \varphi$, as needed.

$\Leftarrow$ We assume that $\{\varphi\}M_{\gtrsim}\{\psi\}$ does not hold, and show that there exists a hard state H and soft state S such that for every game $G \in \mathcal{G}_{(M,H)_{\gtrsim}}(\psi)$ we have that if $|G| = 1$ then $S \notin \mathcal{O}_{\text{SPNE}}(G)$. From our assumption, we have that there exists a hard state H and soft state S for which $H \cup S \models_{\text{prop}} \varphi$ such that either (1) $\neg\exists \alpha \in Act_H : H(\alpha) \cup S \models_{\text{prop}} \psi$, or, (2) $\neg\forall \alpha \in Act_H, \exists S' : H(\alpha) \cup S' \models_{\text{prop}} \psi$ and $S \gtrsim_{ag(H)} S'$. We assume an arbitrary game $G \in \mathcal{G}_{(M,H)_{\gtrsim}}(\psi)$ for which $|G| = 1$ (note that if such a game does not exist, the property we need to show holds trivially) and show that $S \notin \mathcal{O}_{\text{SPNE}}(G)$ must hold:

- If it is the case that $\neg\exists \alpha \in Act_H : H(\alpha) \cup S \models_{\text{prop}} \psi$, we know that $\forall \alpha \in Act_H : f(HH(\alpha)) \neq S$, and thus $S \notin \mathcal{O}_{\text{SPNE}}(G)$.
- If it is the case that $\neg\forall \alpha \in Act_H, \exists S' : H(\alpha) \cup S' \models_{\text{prop}} \psi$ and $S \gtrsim_{ag(H)} S'$, we know that there exists an $\alpha \in Act_H$ such that $f(HH(\alpha)) \succ_{ag(H)} S$, and thus that $S \notin \mathcal{O}_{\text{SPNE}}(G)$.

Since game G was chosen arbitrarily, we know that for all games $G \in \mathcal{G}_{(M,H)_{\gtrsim}}(\psi)$ we have that if $|G| = 1$, then $S \notin \mathcal{O}_{\text{SPNE}}(G)$, as needed.

□

In words, this lemma shows that whenever we have $\{\varphi\}M_{\gtrsim}\{\psi\}$, we have that for every hard state H and soft state S that satisfies φ, that there exists a game $G = ((M, H)_{\gtrsim}, f, T)$ starting from initial hard state H, where f is an arbitrary sequence-based norm from the set $\mathcal{N}_M(\psi)$ and the game tree T is a game tree of length 1 (i.e. the game ends after the performance of a single action), that is, $G \in \mathcal{G}_{(M,H)_{\gtrsim}}(\psi)$ and $|G| = 1$. Moreover, an optimal action in this game realizes outcome S, that is, $S \in \mathcal{O}_{\text{SPNE}}(G)$. Observe that a game triple only states something about the *existence* of a game. It does not state that we have this result for all possible games we can construct based on the postcondition ψ.

To illustrate how game triples encode information about games, we return to the same assertion introduced in Example 4.3, and now illustrate that we can show the validity of this assertion by considering a game.

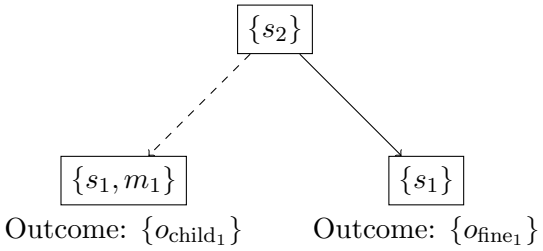


Figure 4.5: This figure illustrates the validity of the game found in Example 4.3. Since it is the case that $\{o_{\text{child}_1}\} \succ_1 \{o_{\text{fine}_1}\}$, agent 1 prefers to go from state $\{s_2\}$ to state $\{s_1, m_1\}$ whilst realizing outcome $\{o_{\text{child}_1}\}$. The dashed transition shows the optimal action.

Example 4.4. *We return to the assertion found in Example 4.3:*

$$\{\phi^{hard}_{\{s_2\}} \wedge \phi^{soft}_{\{o_{\text{child}_1}\}}\} M_{\gtrsim} \{(\phi^{hard}_{\{s_1,m_1\}} \wedge \phi^{soft}_{\{o_{\text{child}_1}\}}) \vee (\phi^{hard}_{\{s_1\}} \wedge \phi^{soft}_{\{o_{\text{fine}_1}\}}\}$$

Consider the game $G = (M_{\gtrsim}, f, T)$, where we let $T = \{\{s_2\}, \{s_2\}\{s_1, m_1\}, \{s_2\}\{s_1\}\}$, and let $\{\{s_2\}\{s_1, m_1\}, \{s_2\}\{s_1\}\}\} \subseteq dom_M(f)$ for which we have $f(\{s_2\}\{s_1, m_1\}) = \{o_{\text{child}_1}\}$ and $f(\{s_2\}\{s_1\}) = \{o_{\text{fine}_1}\}$. Observe that the sequence $\{s_2\}\{s_1, m_1\}$ is the history starting in hard state $\{s_2\}$, and ending in hard state $\{s_1, m_1\}$, and the sequence $\{s_2\}\{s_1\}$ the history starting in hard state $\{s_2\}$ and ending in hard state $\{s_1\}$. This game is drawn in Figure 4.5. We have that $G \in \mathcal{G}_{(M,\{s_2\})_{\gtrsim}}(\psi)$, that $|G| = 1$, and that $\{o_{\text{child}_1}\} \in \mathcal{O}_{SPNE}(G)$, which shows the validity of the game triple.

Not only do these game triples syntactically resemble classical Hoare or Unity triples, they also share a lot of the same properties. One important property is strengthening of the precondition or weakening of the postcondition, which is shown in the next proposition.

Proposition 4.6 (Strengthening and weakening)**.** *Whenever it is the case that $\vDash_{prop} \varphi \to \varphi'$, $\{\varphi'\} M_{\gtrsim} \{\psi'\}$ and $\vDash_{prop} \psi' \to \psi$ we can infer $\{\varphi\} M_{\gtrsim} \{\psi\}$.*

Proof. Assume $\vDash_{\text{prop}} \varphi \to \varphi'$, $\{\varphi'\} M_{\gtrsim} \{\psi'\}$ and $\vDash_{\text{prop}} \psi' \to \psi$, and assume an arbitrary hard state H and soft state S for which $H \cup S \vDash \varphi$. Since $\vDash_{\text{prop}} \varphi \to \varphi'$, we have that $H \cup S \vDash \varphi'$, and thus from $\{\varphi'\} M_{\gtrsim} \{\psi'\}$ we can conclude that (1) $\exists \alpha \in Act_H : H(\alpha) \cup S \vDash_{\text{prop}} \psi'$ and (2) $\forall \alpha \in Act_H, \exists S' : H(\alpha) \cup S' \vDash_{\text{prop}} \psi'$ and $S \gtrsim_{ag(H)} S'$. However, from $\exists \alpha \in Act_H : H(\alpha) \cup S \vDash_{\text{prop}} \psi'$ and $\vDash_{\text{prop}} \psi' \to \psi$, we can conclude (1) $H(\alpha) \cup S \vDash_{\text{prop}} \psi$, and from $\forall \alpha \in Act_H, \exists S' : H(\alpha) \cup S' \vDash_{\text{prop}} \psi'$ and $S \gtrsim_{ag(H)} S'$ and $\vDash_{\text{prop}} \psi' \to \psi$, we can conclude (2) $\forall \alpha \in Act_H, \exists S' : H(\alpha) \cup S' \vDash_{\text{prop}} \psi$ and $S \gtrsim_{ag(H)} S'$. Combining (1) and (2),

and by using the fact that H and S were chosen arbitrarily, we can conclude $\{\varphi\}M_{\gtrsim}\{\psi\}$. □

As mentioned earlier, game triples *generalize* the classical interpretation of a triple we can find in Hoare or Unity logic. That is, whenever in a triple $\{\varphi\}M_{\gtrsim}\{\psi\}$ it is the case that the formulas φ and ψ only range over hard facts, we acquire the standard interpretation of such a triple. This is shown in the next proposition.

Proposition 4.7. *Given a multi-agent preference structure $M_{\gtrsim}$ and two formulas $\varphi, \psi \in \mathcal{L}_{prop}(\Pi_{\text{hard}})$ ranging over hard facts, we have that $\{\varphi\}M_{\gtrsim}\{\psi\}$ iff $\forall H$ for which $H \models_{prop} \varphi : (Act_H \neq \varnothing$ and $\forall \alpha \in Act_H : H(\alpha) \models_{prop} \psi)$.*

Proof. We use the fact that for every formula $\varphi \in \mathcal{L}_{\text{prop}}(\Pi_{\text{hard}})$, hard state $H \in \mathcal{P}(\Pi_{\text{hard}})$ and soft state $S \in \mathcal{P}(\Pi_{\text{soft}})$ it holds that $H \models_{\text{prop}} \varphi \Leftrightarrow H \cup S \models_{\text{prop}} \varphi$. The condition $\exists \alpha \in Act_H : H(\alpha) \cup S \models_{\text{prop}} \psi$ in the definition of $\{\varphi\}M_{\gtrsim}\{\psi\}$ reduces to $\exists \alpha \in Act_H : H(\alpha) \models_{\text{prop}} \psi$, and the condition $\forall \alpha \in Act_H, \exists S' : H(\alpha) \cup S' \models_{\text{prop}} \psi$ and $S \gtrsim_{ag(H)} S'$ reduces to $\forall \alpha \in Act_H : H(\alpha) \models_{\text{prop}} \psi$, since $S \gtrsim_i S$ for any soft state S and agent i. Together, they form the condition $\forall H$ for which $H \models_{\text{prop}} \varphi : (\exists \alpha \in Act_H : H(\alpha) \models_{\text{prop}} \psi$ and $\forall \alpha \in Act_H : H(\alpha) \models_{\text{prop}} \psi)$, which is equivalent to $\forall H$ for which $H \models_{\text{prop}} \varphi : (Act_H \neq \varnothing$ and $\forall \alpha \in Act_H : H(\alpha) \models_{\text{prop}} \psi)$. □

Looking again at the multi-agent system of Solomon's Dilemma, introduced in Section 4.3.1 (p 75), the following assertions are valid:

- $\{s_1\}M_{\gtrsim}\{\neg s_1\}$: If mother 1 has just spoken, then after every possible action that can occur it is the case that mother 1 has not spoken.
- $\{\top\}M_{\gtrsim}\{s_1 \vee s_2\}$: For every hard state after every possible action that can occur, it is the case that either mother 1 or mother 2 has just spoken.

As we have mentioned earlier, game triples are used as the elementary building blocks of the proof system which we will develop later on in this chapter, which allows us to prove more complex properties. The property of 'ensures', based on the similarly named progression property found in the Unity proof-system, is such a property.

4.4.2 Ensures

The ensures property is the fundamental progression property of the original Unity proof system. The statement φ ensures ψ in Unity would assert that once we reach a φ-state, we are either in a ψ-state, or we will eventually end up in a ψ-state, and until we do, φ remains true. This is very closely related to the binary 'until' operator found in the previous chapter, namely φ remains

true until ψ becomes true. In other words, to be in a φ state *ensures* us that we are, or end up, in a ψ-state. The underlying idea of this progression property is that because we do not know which assignment will be executed at what time, progression from φ to ψ is based on the fact that *some* assignment can bring us to the next state, but we do not know when this will occur exactly. In our framework, we want to reason about games. Moreover, there *is* a central notion of control, particularly the agents get to decide an action. Thus, in our framework, progression from φ to ψ means that once we are in a φ-state, we are either already in ψ state, or the *optimal action* for the agent in the φ-state will bring us to a ψ-state. Formally we define them as follows.

Definition 4.16 (Ensures)**.** *Given a multi-agent system* $M = (\Pi, Ags, ag, Act, H_0)$*, the logical language of* ensures *consists of statements of the form* φ *ensures* ψ*, where* φ *and* ψ *are from the set* $\mathcal{L}_{prop}(\Pi)$*. We say that the assertion* φ *ensures* ψ *is* $M_{\gtrsim}$*-valid (valid in the context of* $M_{\gtrsim}$*), notation* $\models_{M_{\gtrsim}} \varphi$ *ensures* ψ*, iff for every hard state* H *and soft state* S *for which* $H \cup S \models_{prop} \varphi$*, there exists a game* $G \in \mathcal{G}_{(M,H)_{\gtrsim}}(\psi)$ *such that* $S \in \mathcal{O}_{SPNE}(G)$ *and* $|G| \leq 1$*.*

In words, we have $\models_{M_{\gtrsim}} \varphi$ ensures ψ if and only if for every hard state H and soft state S that satisfies φ, there exist a game $G = ((M,H)_{\gtrsim}, f, T)$ starting from initial hard state H, where f is an arbitrary sequence-based norm from the set $\mathcal{N}_M(\psi)$ and the game tree T is a game tree of length 0 or 1 (i.e. the game starts in a terminal node, or the game ends after the performance of a single action), that is, $G \in \mathcal{G}_{(M,H)_{\gtrsim}}(\psi)$ and $|G| \leq 1$. Moreover, an optimal action in this game realizes outcome S, that is, $S \in \mathcal{O}_{\text{SPNE}}(G)$. Notice that the latter requirement of $|G| \leq 1$ makes the property of 'Ensures' different than the interpretation we gave to $\{\varphi\} M_{\gtrsim} \{\psi\}$, which had a similar interpretation except that the underlying games were of length 1 exactly.

Note that the property of 'Ensures' (and 'Leads-to' later on) are assertions over arbitrary games that the agents may play, and that these assertions are not related to the *minimal games* that we consider in Section 4.3. Later in this chapter when we return to the topic of mechanism design, we will show how we can use these assertions to reason about properties of the minimal games. Ultimately, we are interested in proving these properties within a proof system, and this proof system only uses game triples as elementary building blocks. More particularly, in this system the game triples are the axioms of the system, and we can infer the 'ensures' properties by applying certain rules of deduction. This will be the topic of the next section.

Proof system for 'Ensures': ENS($M_{\gtrsim}$)

Before we present our proof system, we momentarily want to discuss to the proof system of Unity to reflect on how assertions of the type 'Ensures' are

Axioms	Rules
All valid game triples: $\{\varphi\}M_{\gtrsim}\{\psi\}$	$\dfrac{\{\varphi \wedge \neg\psi\}M_{\gtrsim}\{\psi\}}{\varphi \text{ ensures } \psi}$ (ENS)

Table 4.1: System $\mathbf{ENS}(\mathbf{M}_{\gtrsim})$

proven there. The underlying proof system of this programming language elegantly uses the assumption of fairness to prove these assertions. In this proof system, φ ensures ψ can be deduced if the following two conditions are met:

1. If we are in a $(\varphi \wedge \neg\psi)$-state, it must hold that after execution of *all possible assignments* we end up in a $(\varphi \vee \psi)$-state. This shows a property of *invariance*, once we are in a $(\varphi \wedge \neg\psi)$-state, we will either stay in a $(\varphi \wedge \neg\psi)$-state or go to a ψ-state.

2. There must exist *at least one assignment* that when executed in a $(\varphi \wedge \neg\psi)$-state, will cause the next state to be a ψ-state.

Because we have the property of fairness, we know that eventually the assignment that brings us from a $(\varphi \wedge \neg\psi)$-state to a ψ-state will be executed.

In our model, since for each state it is always the case that the same agent gets to perform an action, we only have to prove that there exists an optimal action that brings us from $(\varphi \wedge \neg\psi)$-state to a ψ-state, and we do not need to show a property of invariance to show the property of 'Ensures'. Formally, given a multi-agent preference structure $M_{\gtrsim}$, the underlying proof system $\mathbf{ENS}(\mathbf{M}_{\gtrsim})$ is shown in Table 4.1. The axioms of this system consists of all valid game triples, and the only rule we have is that we can deduce from $\{\varphi \wedge \neg\psi\}M_{\gtrsim}\{\psi\}$ that φ ensures ψ. We write $\vdash_{M_{\gtrsim}} \Gamma$ to say that a statement Γ is provable in system $\mathbf{ENS}(\mathbf{M}_{\gtrsim})$. We are now ready to show that this proof system is both sound and complete.

Theorem 4.1 (Soundness and completeness of $\mathbf{ENS}(\mathbf{M}_{\gtrsim})$)**.** *The logic of ensures is soundly and completely axiomatized by system* $\mathbf{ENS}(\mathbf{M}_{\gtrsim})$*. That is:*

$$\vdash_{M_{\gtrsim}} \varphi \textit{ ensures } \psi \quad \Leftrightarrow \quad \vDash_{M_{\gtrsim}} \varphi \textit{ ensures } \psi$$

Proof. Since every valid game triple $\{\varphi'\}M_{\gtrsim}\{\psi'\}$ is an axiom of $\mathbf{ENS}(\mathbf{M}_{\gtrsim})$, and the only way to infer φ ensures ψ in this system is by means of such a triple, it suffices to show that:

$$\{\varphi \wedge \neg\psi\}M_{\gtrsim}\{\psi\} \quad \Leftrightarrow \quad \vDash_{M_{\gtrsim}} \varphi \text{ ensures } \psi$$

Soundness follows from the direction left-to-right ($\Rightarrow$) and completeness from the direction right-to-left ($\Leftarrow$):

$\Rightarrow$ Assume $\{\varphi \wedge \neg\psi\} M_{\gtrsim} \{\psi\}$, and assume an arbitrary hard state H and soft state S for which $H \cup S \models_{\text{prop}} \varphi$. We differentiate between the cases where $H \cup S \models_{\text{prop}} \psi$ and where $H \cup S \not\models_{\text{prop}} \psi$:

- If it is the case that $H \cup S \models_{\text{prop}} \psi$, we know that there exists a $G \in \mathcal{G}_{(M,H)_{\gtrsim}}(\psi)$ such that $S \in \mathcal{O}_{\text{SPNE}}(G)$ and $|G| = 0$, namely the game $G = (M_{\gtrsim}, f, T)$ such that $T = \{H\}$ and $f(H) = S$ (this game starts in a terminal node).
- If it is the case that $H \cup S \not\models_{\text{prop}} \psi$, and thus $H \cup S \models_{\text{prop}} \varphi \wedge \neg\psi$, since we have that $\{\varphi \wedge \neg\psi\} M_{\gtrsim} \{\psi\}$ we can apply Lemma 4.1 to derive that there exists a $G \in \mathcal{G}_{(M,H)_{\gtrsim}}(\psi)$ such that $S \in \mathcal{O}_{\text{SPNE}}(G)$ and $|G| = 1$.

Since hard state H and soft state S were chosen arbitrarily, we may conclude that the above construction works for every hard state H and soft state S for which $H \cup S \models_{\text{prop}} \varphi$, and thus that $\models_{M_{\gtrsim}} \varphi$ ensures ψ holds.

$\Leftarrow$ Assume $\models_{M_{\gtrsim}} \varphi$ ensures ψ, and assume an arbitrary hard state H and soft state S for which $H \cup S \models_{\text{prop}} \varphi$ and $H \cup S \not\models_{\text{prop}} \psi$. This implies that for all $f \in \mathcal{N}_M(\psi)$ we have that $f(H) \neq S$, and thus that $\forall G \in \mathcal{G}_{(M,H)_{\gtrsim}}(\psi)$ for which $S \in \mathcal{O}_{\text{SPNE}}(G)$ we have that $|G| \neq 0$, and thus that $|G| = 1$. Since H and S were chosen arbitrarily, this property must hold for every H and S for which $H \cup S \models_{\text{prop}} \varphi$ and $H \cup S \not\models_{\text{prop}} \psi$, i.e. $H \cup S \models_{\text{prop}} \varphi \wedge \neg\psi$. From this and from $\models_{M_{\gtrsim}} \varphi$ ensures ψ we can derive that for every hard state H and soft state S for which $H \cup S \models_{\text{prop}} \varphi \wedge \neg\psi$ there exists a game $G \in \mathcal{G}_{(M,H)_{\gtrsim}}(\psi)$ such that $S \in \mathcal{O}_{\text{SPNE}}(G)$ and $|G| = 1$, which due to Lemma 4.1 implies that $\{\varphi \wedge \neg\psi\} M_{\gtrsim} \{\psi\}$ holds, as needed.

□

The property of 'Ensures' allows us to state assertions that relate to immediate progression, i.e. we can immediately go from a φ-state to a ψ-state, or we are already in a φ-state. The property of 'Leads-to' allows us to state assertions that relate to *eventual* progression, which we will discuss in the next section.

4.4.3 Leads-to

The leads-to property is the fundamental reachability property of the original Unity proof system. The statement φ leads-to ψ in Unity would assert that

once we reach a φ-state, we will *eventually* reach a ψ-state. This is very closely related to the 'diamond' operator found in the previous chapter, namely once φ becomes true, somewhere in the future ψ becomes true. In other words, to be in a φ-state *leads to* a ψ-state. Notice that this property is slightly weaker than the 'Ensures' property, since we drop the restriction that we directly have to go from a φ-state to a ψ-state . Again, in our framework, we want to reason about games, and thus attach a different meaning to this property.

Definition 4.17 (Leads-to)**.** *Given a multi-agent system* $M = (\Pi, Ags, ag, Act, H_0)$*, the logical language of* leads-to *consists of statements of the form* φ *leads-to* ψ*, where* φ *and* ψ *are from the set* $\mathcal{L}_{prop}(\Pi)$*. We say that the assertion* φ *leads-to* ψ *is* $M_{\gtrsim}$*-valid (valid in the context of* $M_{\gtrsim}$*), notation* $\models_{M_{\gtrsim}} \varphi$ *leads-to* ψ*, iff for every hard state* H *and soft state* S *for which* $H \cup S \models_{prop} \varphi$*, there exists a game* $G \in \mathcal{G}_{(M,H)_{\gtrsim}}(\psi)$ *such that* $S \in \mathcal{O}_{SPNE}(G)$*.*

In words, we have $\models_{M_{\gtrsim}} \varphi$ leads-to ψ if and only if for every hard state H and soft state S that satisfies φ, there exists a game $G = ((M,H)_{\gtrsim}, f, T)$ starting from initial hard state H, where f is an arbitrary sequence-based norm from the set $\mathcal{N}_M(\psi)$ and the game tree T is a game tree of arbitrary (but finite) length, that is, $G \in \mathcal{G}_{(M,H)_{\gtrsim}}(\psi)$. Moreover, a sub-game perfect Nash equilibrium strategy profile in this game realizes outcome S, that is, $S \in \mathcal{O}_{\text{SPNE}}(G)$. Notice that this time around, we have no requirements on the game length. This was previously the case with the interpretation we gave to $\{\varphi\}M_{\gtrsim}\{\psi\}$, which required that the underlying games were of length 1 exactly, and with φ ensures ψ, which required that the underlying games were of length 0 or 1 exactly.

We are now ready to introduce the complete proof system in the next section, which allows us to deduce both 'Ensures' and 'Leads-to' from only game triples.

Proof system for 'Leads-to': $\mathbf{LT}(\mathbf{M}_{\gtrsim})$

In the original proof system of Unity, derivation of φ leads-to ψ could be achieved by taking the transitive closure of the 'Ensures' property. Our proof system works in a similar manner, and is shown in Table 4.2. This proof system consists of all the rules and axioms of system $\mathbf{ENS}(\mathbf{M}_{\gtrsim})$, but introduces two additional rules (LT1) and (LT2) to derive statements of the form φ leads-to ψ.

Before we make any soundness and completeness claims regarding system $\mathbf{LT}(\mathbf{M}_{\gtrsim})$, we need to prove several smaller lemmas, in particular Lemma 4.2 (strengthening of the precondition) and Lemma 4.3 (composition of preconditions). These lemmas are all related to the completeness of this system, since they make claims about things that are provable within the system. Later on, we can use these lemmas to show full completeness.

Rules
$\dfrac{\varphi \text{ ensures } \psi}{\varphi \text{ leads-to } \psi}$ (LT1) $\qquad$ $\dfrac{\varphi \text{ leads-to } \psi' \qquad \psi' \text{ leads-to } \psi}{\varphi \text{ leads-to } \psi}$ (LT2)
+ All axioms and rules from system $\mathbf{ENS}(\mathbf{M}_{\gtrsim})$

Table 4.2: System $\mathbf{LT}(\mathbf{M}_{\gtrsim})$

Lemma 4.2 (Strengthening of the precondition)**.** *Given a multi-agent preference structure* $M_{\gtrsim}$*, if we have* $\vDash_{prop} \varphi \to \varphi'$ *and* $\vdash_{M_{\gtrsim}} \varphi'$ *leads-to* ψ*, we can derive* $\vdash_{M_{\gtrsim}} \varphi$ *leads-to* ψ*.*

Proof. Since there exists no hard state H and soft state S such that $H \cup S \vDash_{\text{prop}} \varphi \wedge \neg\varphi'$, we obtain the general validity of $\{\varphi \wedge \neg\varphi'\}M_{\gtrsim}\{\varphi'\}$, and is thus an axiom of $\mathbf{ENS}(\mathbf{M}_{\gtrsim})$. From this we can apply rule (ENS) and (LT1) to acquire $\vdash_{M_{\gtrsim}} \varphi$ leads-to φ', and together with $\vdash_{M_{\gtrsim}} \varphi'$ leads-to ψ we can apply rule (LT2) to acquire $\vdash_{M_{\gtrsim}} \varphi$ leads-to ψ. □

Note that an important consequence of this lemma is that the proof system is closed under syntactic rewriting of the precondition. For example, if $\vdash_{M_{\gtrsim}} \varphi$ leads-to ψ and φ' is logically equivalent to φ (but syntactically different), we have the propositional validity $\vDash_{\text{prop}} \varphi' \to \varphi$, and thus from the lemma $\vdash_{M_{\gtrsim}} \varphi'$ leads-to ψ. Note that we can construct a similar argument that shows that the proof system is closed under weakening of the postcondition.

Lemma 4.3 (Composition of preconditions)**.** *Given a multi-agent preference structure* $M_{\gtrsim}$*, from* $\vdash_{M_{\gtrsim}} \varphi$ *leads-to* ψ *and* $\vdash_{M_{\gtrsim}} \varphi'$ *leads-to* ψ*, we can infer* $\vdash_{M_{\gtrsim}} (\varphi \vee \varphi')$ *leads-to* ψ*.*

Proof. Our first step is to show that from $\vdash_{M_{\gtrsim}} \varphi$ ensures ψ we can infer $\vdash_{M_{\gtrsim}} (\varphi \vee \varphi')$ ensures $(\psi \vee \varphi')$ (for any formula φ'). We have:

$$\vdash_{M_{\gtrsim}} \varphi \text{ ensures } \psi \;\Rightarrow\; \{\varphi \wedge \neg\psi\}M_{\gtrsim}\{\psi\} \overset{!}{\Rightarrow}$$
$$\{(\varphi \vee \varphi') \wedge \neg(\psi \vee \varphi')\}M_{\gtrsim}\{\psi \vee \varphi'\} \;\Rightarrow \vdash_{M_{\gtrsim}} (\varphi \vee \varphi') \text{ ensures } (\psi \vee \varphi')$$

The inference above marked with $\overset{!}{\Rightarrow}$ uses Proposition 4.6 (strengthening and weakening) together with the fact that:

$$\vDash_{\text{prop}} ((\varphi \vee \varphi') \wedge \neg(\psi \vee \varphi')) \to (\varphi \wedge \neg\psi)$$

From $\vdash_{M_{\gtrsim}} \varphi$ leads-to ψ, we know that there exists a chain of formulas $\varphi_0 \ldots \varphi_n$ for a certain $n \geq 1$, where φ_0 is equal to φ and φ_n is equal to ψ, such that for

all t, $0 < t \leq n$:

$$\vdash_{M_{\gtrsim}} \varphi_{t-1} \text{ ensures } \varphi_t$$

And likewise from $\vdash_{M_{\gtrsim}} \varphi'$ leads-to ψ that there exists a chain of formulas $\varphi'_0 \dots \varphi'_{n'}$ for a certain $n' \geq 1$, where φ'_0 is equal to φ' and $\varphi'_{n'}$ is equal to ψ, such that for all t, $0 < t \leq n'$:

$$\vdash_{M_{\gtrsim}} \varphi'_{t-1} \text{ ensures } \varphi'_t$$

From this we can infer:

$$\vdash_{M_{\gtrsim}} (\varphi_0 \vee \varphi'_0) \text{ ensures } (\varphi_1 \vee \varphi'_0), \dots, \vdash_{M_{\gtrsim}} (\varphi_{n-1} \vee \varphi'_0) \text{ ensures } (\varphi_n \vee \varphi'_0)$$

$$\vdash_{M_{\gtrsim}} (\varphi'_0 \vee \varphi_n) \text{ ensures } (\varphi'_1 \vee \varphi_n), \dots, \vdash_{M_{\gtrsim}} (\varphi'_{n'-1} \vee \varphi_n) \text{ ensures } (\varphi'_{n'} \vee \varphi_n)$$

From multiple applications of rule (LT1) and (LT2) we are able to infer $\vdash_{M_{\gtrsim}} (\varphi_0 \vee \varphi'_0)$ leads-to $(\varphi'_{n'} \vee \varphi_n)$. By recalling that φ_0 is equal to φ, φ'_0 is equal to φ', and that both φ_n and $\varphi'_{n'}$ are equal to ψ, we get $\vdash_{M_{\gtrsim}} (\varphi \vee \varphi')$ leads-to $(\psi \vee \psi)$. Finally, since trivially $\vdash_{M_{\gtrsim}} (\psi \vee \psi)$ leads-to ψ, we get from one more application of (LT2) the desired result $\vdash_{M_{\gtrsim}} (\varphi \vee \varphi')$ leads-to ψ. □

Definition 4.18 (Instances of games)**.** *Given $\vDash_{M_{\gtrsim}} \varphi$ leads-to ψ and given an arbitrary hard state H and soft state S for which $H \cup S \vDash_{prop} \varphi$, let $G' = ((M,H)_{\gtrsim}, f', T')$ be the game for which $G' \in \mathcal{G}_{(M,H)_{\gtrsim}}(\psi)$ and for which there exists a SPNE strategy profile σ' such that $f(run(\sigma')) = S$. We write $G_\psi[H,S]$ to refer to this G', $f_\psi[H,S]$ to refer to this f', $T_\psi[H,S]$ to refer to this T', and finally $\sigma_\psi[H,S]$ to refer to this σ'. Whenever there exists multiple combinations of games, enforcement norms and SPNE strategy profiles that satisfy this criterion for a given hard-soft state pair, we may choose one arbitrarily.*

This notation will be useful to explicitly refer to the games and strategies that a leads-to assertion implicitly encodes, and will be used in the upcoming proof. For example, given a multi-agent system preference structure $M_{\gtrsim}$, whenever we have $\vDash_{M_{\gtrsim}} \varphi$ leads-to ψ, we can infer the existence of the following games:

$$\{G_\psi[H,S] \mid H \in \mathcal{P}(\Pi_{\text{hard}}), S \in \mathcal{P}(\Pi_{\text{soft}}), H \cup S \vDash_{\text{prop}} \varphi\}$$

Using this newly acquired notation, we are now ready to prove soundness and completeness of system $\mathbf{LT}(\mathbf{M}_{\gtrsim})$. Note that the proof is quite lengthy, which is why each of the steps start with a general proof strategy for the reader not concerned with the specific technicalities.

Theorem 4.2 (Soundness and completeness of $\mathbf{LT}(\mathbf{M}_{\gtrsim})$)**.** *The logic of leads-to is (soundly and completely) axiomatized by system $\mathbf{LT}(\mathbf{M}_{\gtrsim})$. That is:*

$$\vdash_{M_{\gtrsim}} \varphi \textit{ leads-to } \psi \quad \Leftrightarrow \quad \vDash_{M_{\gtrsim}} \varphi \textit{ leads-to } \psi$$

Proof. We show soundness ($\Rightarrow$) and completeness ($\Leftarrow$) separately.

$\Rightarrow$ Soundness can be shown by showing soundness of the rules (LT1) and (LT2), since soundness of the remaining rules follows from Theorem 4.1. Soundness of inference rule (LT1) is trivial, since φ leads-to ψ is just a generalization of φ ensures ψ where we drop the restriction on the size of the game. To show soundness of rule (LT2), we have to show that:

$$\vDash_{M_{\gtrsim}} \varphi \text{ leads-to } \psi' \quad \text{and} \quad \vDash_{M_{\gtrsim}} \psi' \text{ leads-to } \psi \quad \Rightarrow \quad \vDash_{M_{\gtrsim}} \varphi \text{ leads-to } \psi$$

The general idea of this proof is that statement $\vDash_{M_{\gtrsim}} \varphi$ leads-to ψ' encodes the existence of certain games, and that $\vDash_{M_{\gtrsim}} \psi'$ leads-to ψ encodes the existence of certain games. Combining these games in certain relevant ways, we can show the existence of certain (larger) composed games, which in turn entail the validity $\vDash_{M_{\gtrsim}} \varphi$ leads-to ψ. The complete construction is shown below.

Let us first assume an arbitrary hard state H_0 and soft state S_0 for which $H_0 \cup S_0 \vDash_{\text{prop}} \varphi$. From $\vDash_{M_{\gtrsim}} \varphi$ leads-to ψ', let $G_0 = ((M, H_0)_{\gtrsim}, f_0, T_0) \in \mathcal{G}_{(M,H_0)_{\gtrsim}}(\psi')$ be such that $S_0 \in \mathcal{O}_{\text{SPNE}}(G_0)$, and let $\sigma_0 \in \Sigma_{\text{SPNE}}(G_0)$ be the SPNE strategy profile for which $run(\sigma_0) = S_0$. Given an arbitrary terminal history $h \in \hat{T}_0$ (recall from Definition 4.9 that for an arbitrary game tree T, the set $\hat{T}$ denotes the terminal histories), we have that $H_h \cup f_0(h) \vDash \psi'$. But since we have that $\vDash_{M_{\gtrsim}} \psi'$ leads-to ψ from our assumption, we use the notation $G_\psi[H_h, f_0(h)]$, $T_\psi[H_h, f_0(h)]$, $f_\psi[H_h, f_0(h)]$ and $\sigma_\psi[H_h, f_0(h)]$ from Definition 4.18 to refer to these corresponding games, game trees, sequence-based norms and SPNE strategy profiles respectively. We create the compound game $G^C = ((M, H_0)_{\gtrsim}, f^C, T^C)$, where T^C is defined as follows:

$$T^C := T_0 \cup \{h + h' \mid h \in \hat{T}_0, \;\; h' \in T_\psi[H_h, f_0(h)]\}$$

Observe that the set of terminal histories from this normative game tree $\hat{T}^C$ are all compound histories. For these terminal histories $(h + h') \in \hat{T}^C$ (where $h \in \hat{T}_0$ and $h' \in \hat{T}_\psi[H_h, f_0(h)]$) we let the function f^C be defined as follows:

$$f^C(h + h') := f_\psi[H_h, f_0(h)](h')$$

For any non-terminal history $h^C \in T^C \backslash \hat{T}^C$, we can let $f^C(h^C)$ be any value since it has no influence on the game. This concludes the construction of the compound game, and this construction is also illustrated in Figure 4.6 where we create from initial game G_0 and several games $G_\psi[H_h, f_0(h)]$ (where $h \in \hat{T}_0$ is a terminal history in the initial game G_0) the compound game shown at the bottom of the picture. It is clear

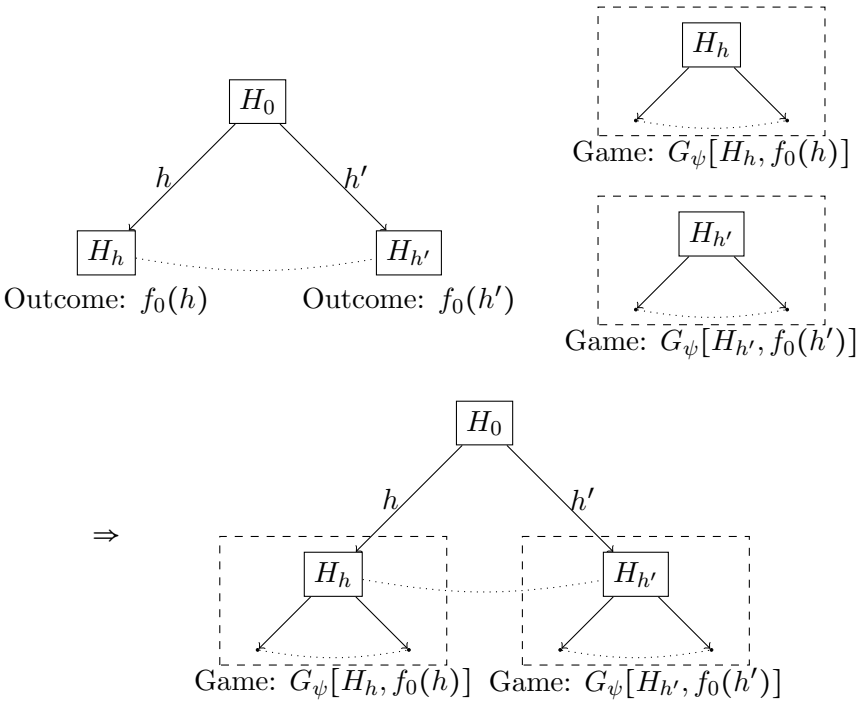


Figure 4.6: Figure showing how we can create a compound game (shown at the bottom) from the game on the top left and the games on the right. The construction is formally described in Theorem 4.2 (soundness).

4

from this construction that every history $h^C \in T^C$ from the game tree of this compound game is either in $T_0 \backslash \hat{T}_0$, or if it is not it is a compound history $h^C = h + h'$ for which $h \in \hat{T}_0$ and $h' \in T_\psi[H_h, f_0(h)]$. For this particular game G^C we have that $G^C \in \mathcal{G}_{(M,H_0)_\trianglerighteq}(\psi)$, so if we can show that $S_0 \in \mathcal{O}_{\text{SPNE}}(G^C)$, we can conclude from the fact that H_0 and S_0 were chosen arbitrarily that this construction is possible for every H and S for which $H \cup S \vDash_{\text{prop}} \varphi$, and thus that $\vDash_{M_\trianglerighteq} \varphi$ leads-to ψ, as needed.

Secondly, to show that $S_0 \in \mathcal{O}_{\text{SPNE}}(G^C)$, we construct a sub-game perfect Nash equilibrium strategy profile $\sigma^C \in \Sigma_{\text{SPNE}}(G^C)$ for which it holds that $f^C(run(\sigma^C)) = S_0$. We will treat a strategy profile as simply a single function mapping histories to actions, even though it is actually a tuple of strategies, namely one for each agent. We can make this simplification since at any moment in the game there is only a single agent that gets to decide on an action, so there is no confusion as to which action belongs to which agent. Given a history $h^C \in T^C$, if $h^C \in T_0 \backslash \hat{T}_0$, we let $\sigma^C(h^C) = \sigma_0(h^C)$. If $h^C \notin T_0 \backslash \hat{T}_0$, we know that $h^C = h + h'$ is a compound history, for which we let $\sigma^C(h^C) = \sigma_\psi[H_h, f_0(h)](h')$.

This completely specifies strategy profile σ^C. We now have to show that (1) $f^C(run(\sigma^C)) = S_0$ and (2) $\sigma^C \in \Sigma_{\text{SPNE}}(G^C)$, which we will do individually:

(1) To show $f^C(run(\sigma^C)) = S_0$, let $h_0 = run(\sigma_0)$, and observe the following:

$$run(\sigma^C) = h_0 + run(\sigma_\psi[H_{h_0}, f_0(h_0)])$$

Moreover, observe that by Definition 4.18 we have for every terminal history $h \in \hat{T}_0$ the following:

$$f_\psi[H_h, f_0(h)](run(\sigma_\psi[H_h, f_0(h)]) = f_0(h)$$

Which tells us that the Sub-game Perfect Nash Equilibrium strategy profile of the game $G_\psi[H_h, f_0(h)]$ will indeed realize outcome $f_0(h)$. By construction of the compound game we can now conclude the following:

$$\begin{aligned} f^C(run(\sigma^C)) &= f^C(h_0 + run(\sigma_\psi[H_{h_0}, f_0(h_0)])) \\ &= f_\psi[H_{h_0}, f_0(h_0)](run(\sigma_\psi[H_{h_0}, f_0(h_0)])) \\ &= f_0(h_0) \\ &= S_0 \end{aligned}$$

Which is what we needed to show.

(2) To show $\sigma^C \in \Sigma_{\text{SPNE}}(G^C)$, observe that for every history $h^C \in T^C$ if it holds that if $h^C \in T_0 \backslash \hat{T}_0$, then $\sigma^C(h^C) = \sigma_0(h^C)$, and in both sub-games (a) G^C starting from h^C and (b) G_0 starting from h^C the same outcomes are reached at the end of the game by following σ^C or σ_0 respectively. Thus, action $\sigma^C(h^C)$ must constitute a sub-game perfect equilibrium since $\sigma_0(h)$ does. If it holds that if $h^C \notin T_0 \backslash \hat{T}_0$, then $h^C = h + h'$ such that $\sigma^C(h^C) = \sigma_\psi[H_h, f_0(h)](h')$, and in both sub-games (a) G^C starting from h^C and (b) $G_\psi[H_h, f_0(h)]$ starting from h' the same outcomes are reached at the end of the game by following σ^C or $\sigma_\psi[H_h, f_0(h)]$ respectively. Thus, action $\sigma^C(h^C)$ must constitute a sub-game perfect equilibrium since $\sigma_\psi[H_h, f_0(h)](h')$ does.

This allows us to conclude that $S_0 \in \mathcal{O}_{\text{SPNE}}(G^C)$. Since H_0 and S_0 were chosen arbitrarily, we may conclude that the above construction works for every hard state H and soft state S for which $H \cup S \models_{\text{prop}} \varphi$, and thus that $\models_{M_\gtrsim} \varphi$ leads-to ψ, as needed.

$\Leftarrow$ Completeness follows if we can show that from the assumption that $\models_{M_\gtrsim} \varphi$ leads-to ψ, we can infer using the axioms and rules from system

$\mathbf{LT}(\mathbf{M}_{\gtrsim})$ that $\vdash_{M_{\gtrsim}} \varphi$ leads-to ψ. Assume $\vDash_{M_{\gtrsim}} \varphi$ leads-to ψ, and let H and S be the hard and soft state for which $H \cup S \vDash_{\text{prop}} \varphi$. Moreover, let game $G = ((M,H)_{\gtrsim}, f, T) \in \mathcal{G}_{(M,H)_{\gtrsim}}(\psi)$ for which $S \in \mathcal{O}_{\text{SPNE}}(G)$, and let $\sigma \in \Sigma_{\text{SPNE}}(G)$ be the sub-game perfect equilibrium strategy profile that realizes outcome S, i.e. $f(run(\sigma)) = S$. The proof strategy we use is by using the existence of game G, we can look at every possible stage in this game, translate this to a relevant game of length 1, and translate this to the corresponding game triple which are the axioms of our proof system. By using these game triples, we can retrace the same steps that take place in the game, only now in our proof system by linking these game triples together to form more elaborate ensures and leads-to assertions. At the end of this proof, we show that by repeating this process for all hard and soft states that satisfy φ, we end up with $\vdash_{M_{\gtrsim}} \varphi$ leads-to ψ, as needed.

We start by defining P as a list of all histories from T sorted by post-order traversal (where the order in which the children are visited can be chosen arbitrarily). That is, $P = h_0 \dots h_n$, where h_0 is a terminal history and h_n the initial history, which is thus equal to H. We know that this list is finite, since the game tree T is finite. For every i $(0 \leq i \leq n)$ we let H_i be defined as H_{h_i}, and S_i be defined as $f(h_i + run(\sigma_{h_i}))$, where σ_{h_i} is equal to strategy σ, except restricted to the sub-game starting from history h_i. In other words, S_i represents the outcome of the game would we follow the SPNE strategy σ once we arrive at history h_i in T. We define the following formulas (where $0 < i \leq n$):

$$\psi_0 := \psi, \quad \text{and,} \quad \psi_i := (\phi^{\text{hard}}_{H_i} \wedge \phi^{\text{soft}}_{S_i}) \vee \psi_{i-1}$$

We now claim that the following holds (where $0 < i \leq n$):

$$\vdash_{M_{\gtrsim}} \{\psi_i \wedge \neg\psi_{i-1}\} M_{\gtrsim} \{\psi_{i-1}\}$$

Which is a formal way of stating that if we are at an arbitrary point h_i in the game tree T, then if we would consider the game which immediately ends after a single action has taken place and which assigns the outcomes that would have been received if the game would have continued, but at which point we would just follow the SPNE strategy σ. Then for this game, S_i would be the most preferred outcome, which indeed has to be the case since σ is sub-game perfect. Notice that this encodes the procedure of backwards induction, which we typically perform in order to show when, in an extensive game, a strategy profile is sub-game perfect. By applying rule (ENS) and (LT1), this allows us to conclude (where $0 < i \leq n$):

$$\vdash_{M_{\gtrsim}} \psi_i \text{ ensures } \psi_{i-1}, \quad \text{and,} \quad \vdash_{M_{\gtrsim}} \psi_i \text{ leads-to } \psi_{i-1}$$

And, by repeatedly applying (LT2), we get $\vdash_{M_\gtrsim} \psi_n$ leads-to ψ_0. For initial hard state H and soft state S, since ψ_n is equal to $(\phi_H^{\text{hard}} \wedge \phi_S^{\text{soft}}) \vee \psi_{n-1}$ we have the following propositional tautology:

$$\vDash_{\text{prop}} (\phi_H^{\text{hard}} \wedge \phi_S^{\text{soft}}) \rightarrow \psi_n$$

This allows us to use Lemma 4.2 (strengthening of the precondition) to obtain the important result that $\vdash_{M_\gtrsim} (\phi_H^{\text{hard}} \wedge \phi_S^{\text{soft}})$ leads-to ψ.

We can now make the observation that since H and S were chosen arbitrarily, we can repeat the same construction for all H_0,S_0, ..., H_m,S_m $(m \geq 0)$ for which $H_i \cup S_i \vDash_{\text{prop}} \varphi$ $(0 \leq i \leq m)$ and conclude $\vdash_{M_\gtrsim} (\phi_{H_i}^{\text{hard}} \wedge \phi_{S_i}^{\text{soft}})$ leads-to ψ. Using Lemma 4.3 (composition of preconditions) repeatedly, we can combine all these statements into the following statement:

$$\vdash_{M_\gtrsim} ((\phi_{H_0}^{\text{hard}} \wedge \phi_{S_0}^{\text{soft}}) \vee \cdots \vee (\phi_{H_m}^{\text{hard}} \wedge \phi_{S_m}^{\text{soft}})) \text{ leads-to } \psi$$

Since the precondition of this assertion is logically equivalent to φ, we have the propositional validity $\vDash_{\text{prop}} \varphi \rightarrow ((\phi_{H_0}^{\text{hard}} \wedge \phi_{S_0}^{\text{soft}}) \vee \cdots \vee (\phi_{H_m}^{\text{hard}} \wedge \phi_{S_m}^{\text{soft}}))$, which allows us to use Lemma 4.2 to infer $\vdash_{M_\gtrsim} \varphi$ leads-to ψ, as needed.

□

The consequence of this result is that φ leads-to ψ is valid if and only if we can construct a finite binary inference tree, which the next proposition shows.

Theorem 4.3. *$\vDash_{M_\gtrsim} \varphi$ leads-to ψ if and only if we can construct a finite binary inference tree with the following two properties:*

1. *The root node is labelled "φ leads-to ψ" and the leaf nodes are labelled with a game triple, and;*
2. *If a node is labelled Γ and its children are labelled $A_0, \ldots, A_n$, then $A_0, \ldots, A_n \vdash_{M_\gtrsim} \Gamma$.*

Proof. Follows from Theorem 4.2. □

Such a tree can be useful to compactly present the inferences needed to arrive at the desired conclusion, and we will construct such a tree at the end of this chapter to solve Solomon's Dilemma.

Although we have shown soundness and completeness for our system of ensures and leads-to, one important issue that remains is whether these systems are *decidable*. This will be the topic of the next section.

4.4.4 Decidability

The question of decidability in our framework is whether there exists an effective procedure that decides, given an arbitrary assertion from the language of ensure or leads-to, whether this formula is a member of our theory or not. More specifically, given a multi-agent preference structure $M_{\succsim}$, does there exists an *effective procedure* that answers *yes* whenever $\vdash_{M_{\succsim}} \varphi$ leads-to ψ (alternatively $\vdash_{M_{\succsim}} \varphi$ ensures ψ), and answers *no* otherwise. Decidability is not at all obvious in this framework, since the games we consider can be *arbitrarily large*, i.e. we cannot simply enumerate every possible game tree that the agents may play since this set can be infinite. However, as we will show, we have the result that our system is indeed decidable. This result is based on the following two important observations:

1. The pre- and post- conditions of our game-triples, ensures assertions and leads-to assertions are all formulas of propositional logic, for which *decidability is already established.*

2. Although the set of possible game trees can be infinitely large, the set of possible hard and soft states is always *finite* since the set of atomic propositions is *finite.*

We have the following result.

Theorem 4.4 (Decidability)**.** *Given a multi-agent preference structure $M_{\succsim}$, system* $\mathbf{LT}(\mathbf{M}_{\succsim})$ *and* $\mathbf{ENS}(\mathbf{M}_{\succsim})$ *are* decidable.

Proof. Let $\varphi \in \mathcal{L}_{\text{prop}}(\Pi)$ be an arbitrary propositional formula. We define $[\varphi]_{M_{\succsim}} \subseteq \mathcal{P}(\Pi_{\text{hard}}) \times \mathcal{P}(\Pi_{\text{soft}})$ as the *extension* of φ, which are all the hard-soft state pairs that satisfy this formula. That is:

$$[\varphi]_{M_{\succsim}} := \{(H, S) \in \mathcal{P}(\Pi_{\text{hard}}) \times \mathcal{P}(\Pi_{\text{soft}}) \mid H \cup S \vDash_{\text{prop}} \varphi\}$$

Observe that this set can be effectively computed, since the language of propositional logic is decidable, and since the number of hard-soft state pairs is at most $2^{|\Pi|}$ (which is equal to $|[\top]_{M_{\succsim}}|$). We define the following function, called the *weakest pre-condition function*, for game triples, where *Post* is an arbitrary finite set of hard-soft state pairs, and this function returns the largest set of hard-soft states satisfying the precondition of such a game triple:

$$\begin{aligned} &wpre_{\text{game}}(M_{\succsim}, Post) := \{(H, S) \in [\top]_{M_{\succsim}} \mid \\ &\quad \exists \alpha \in Act_H : (H(\alpha), S) \in Post, \text{ and,} \\ &\quad \forall \alpha \in Act_H, \exists (H', S') \in Post : H' = H(\alpha) \text{ and } S \succsim_{ag(H)} S' \\ &\} \end{aligned}$$

The reason we call this function the weakest pre-condition function for game triples is because it has the following two properties:

1. If $[\varphi]_{M_\gtrsim} = wpre_{\text{game}}(M_\gtrsim, [\psi]_{M_\gtrsim})$, then $\{\varphi\}M_\gtrsim\{\psi\}$, and;
2. If $\{\varphi'\}M_\gtrsim\{\psi\}$, then $[\varphi']_{M_\gtrsim} \subseteq wpre_{\text{game}}(M_\gtrsim, [\psi]_{M_\gtrsim})$.

It is clear from Definition 4.15 that this function indeed returns the weakest pre-condition, and it is also clear that this function can be effectively computed since $Post$, $[\top]_{M_\gtrsim}$ and the set of actions Act are finite. From the rules and axioms of system $\mathbf{ENS}(\mathbf{M}_\gtrsim)$, we can derive the following weakest pre-condition function:

$$wpre_{\text{ENS}}(M_\gtrsim, Post) := Post \cup wpre_{\text{game}}(M_\gtrsim, Post)$$

Again this function can be effectively computed, since $Post$ is finite and since the result $wpre_{\text{game}}(M_\gtrsim, Post)$ can be effectively computed. Consider the following sequence $\left(wpre^k_{\text{LT}}(M_\gtrsim, Post)\right)^\infty_{k=0}$ shown below:

$$\begin{aligned}
wpre^0_{\text{LT}}(M_\gtrsim, Post) &= wpre_{\text{ENS}}(M_\gtrsim, Post)\\
wpre^1_{\text{LT}}(M_\gtrsim, Post) &= wpre_{\text{ENS}}(M_\gtrsim, wpre^0_{\text{LT}}(M_\gtrsim, Post))\\
&\ldots = \ldots\\
wpre^k_{\text{LT}}(M_\gtrsim, Post) &= wpre_{\text{ENS}}(M_\gtrsim, wpre^{k-1}_{\text{LT}}(M_\gtrsim, Post))\\
&\ldots = \ldots
\end{aligned}$$

For this particular sequence, we have the following:

$$\begin{aligned}
wpre^0_{\text{LT}}(M_\gtrsim, Post) \subseteq wpre^1_{\text{LT}}(M_\gtrsim, Post) \quad &\subseteq \ldots \subseteq\\
wpre^{k-1}_{\text{LT}}(M_\gtrsim, Post) \subseteq wpre^k_{\text{LT}}(M_\gtrsim, Post) \quad &\subseteq \ldots
\end{aligned}$$

Moreover, since the following also holds:

$$\forall n \geq 0 : wpre^n_{\text{LT}}(M_\gtrsim, Post) \subseteq [\top]_{M_\gtrsim}$$

Meaning that $[\top]_{M_\gtrsim}$ is an upper bound for this sequence, we know that there exists a $0 \leq n^* \leq |[\top]_{M_\gtrsim}|$ for which this sequence has converged to a *fixed point*, that is, for which the following holds:

$$wpre^{n^*}_{\text{LT}}(M_\gtrsim, Post) = wpre^{n^*+1}_{\text{LT}}(M_\gtrsim, Post)$$

We define $wpre^*_{\text{LT}}(M_\gtrsim, Post)$ as this fixed point. That is, given this n^*, we define it as follows:

$$wpre^*_{\text{LT}}(M_\gtrsim, Post) := wpre^{n^*}_{\text{LT}}(M_\gtrsim, Post)$$

Note again that this function can be effectively computed, since n^* is at most $|[\top]_{M_\gtrsim}|$ (which is equal to $2^{|\Pi|}$) and since $wpre_{\text{ENS}}$ can be effectively computed. Decidability now follows from the following correspondence:

$$\begin{aligned} \vdash_{M_\gtrsim} \{\varphi\} M_\gtrsim \{\psi\} &\quad\Leftrightarrow\quad [\varphi]_{M_\gtrsim} \subseteq wpre_{\text{game}}(M_\gtrsim, [\psi]_{M_\gtrsim}) \\ \vdash_{M_\gtrsim} \varphi \text{ ensures } \psi &\quad\Leftrightarrow\quad [\varphi]_{M_\gtrsim} \subseteq wpre_{\text{ENS}}(M_\gtrsim, [\psi]_{M_\gtrsim}) \\ \vdash_{M_\gtrsim} \varphi \text{ leads-to } \psi &\quad\Leftrightarrow\quad [\varphi]_{M_\gtrsim} \subseteq wpre^*_{\text{LT}}(M_\gtrsim, [\psi]_{M_\gtrsim}) \end{aligned}$$

All of which can be effectively computed, as argued above. This concludes the proof. □

As mentioned before, the property of 'Ensures' and 'Leads-to' are assertions over arbitrary games that the agents may play, and that these assertions are not related to the *minimal games* that we consider in Section 4.3. Thus, in order to use this proof system for mechanism design, we need to show how we can use it to prove properties of minimal games, and not just arbitrary-length games. This will be the topic of the next section.

4.5 Proof system for implementability

In this section, we show how we can use our proof system to reason about whether a norm (partially, weakly, fully) implements a social choice rule. Since mechanism design as introduced in Section 4.3 was only concerned with minimal games, and our proof system was able to prove assertions about arbitrary-length games, we need some way of applying our proof system to acquire results about minimal games. The approach we consider in this chapter is to apply a structural update to the multi-agent system, and if we use our proof system on this updated multi-agent system, we only acquire results about minimal games. The underlying idea is that given a state-based norm ψ, the structural update removes all out-going actions from any hard-state H if it is the case that this norm assigns an outcome to H. This update is formally defined as follows.

Definition 4.19. *Given a normative multi-agent system $M = (\Pi, Ags, ag, Act, H_0)$ and a state-based norm $\psi \in \mathcal{L}_{prop}(\Pi)$, we define M^ψ as the structural update $(\Pi, Ags, ag, Act', H_0)$ such that*

$$Act' = \{(\varphi \wedge \neg \left(\bigvee_{H \in Dom} \phi_H^{hard} \right), \Pi^+_{\text{hard}}, \Pi^-_{\text{hard}}) \mid (\varphi, \Pi^+_{\text{hard}}, \Pi^-_{\text{hard}}) \in Act\},$$
$$\text{where } Dom = \{H_h \mid h \in dom_M(\psi)\}$$

Once we acquire for a given multi-agent system M and state-based norm ψ the updated multi-agent system M^ψ, we know that for any history $h \in \mathcal{H}_{M^\psi}$

such that $h \in dom_{M^\psi}(\psi)$, the set Act_{H_h} is empty, i.e. there exists no applicable actions. In particular, we have the following result.

Proposition 4.8. *Given a multi-agent system preference structure $M_\succsim$ and a state-based norm ψ, we have:*

$$\mathcal{G}^{min}_{M_\succsim}(\psi) = \mathcal{G}_{M^\psi_\succsim}(\psi)$$

Proof. The first observation we make is that due to Definition 4.7 we have that $\mathcal{N}_M(\psi) = \{f\}$ where f is an arbitrary enforcement norm which we will explicitly refer to throughout this proof. We can trivially conclude that $\mathcal{T}^{min}_{M^\psi}(f) = \mathcal{T}^{min}_{M}(f)$. We show $\mathcal{G}^{min}_{M_\succsim}(\psi) \subseteq \mathcal{G}_{M^\psi_\succsim}(\psi)$ and $\mathcal{G}^{min}_{M_\succsim}(\psi) \supseteq \mathcal{G}_{M^\psi_\succsim}(\psi)$ separately.

- $\subseteq$ To show $\mathcal{G}^{min}_{M_\succsim}(\psi) \subseteq \mathcal{G}_{M^\psi_\succsim}(\psi)$, we have to show that $\mathcal{T}^{min}_{M}(f) \subseteq \mathcal{T}_{M^\psi}(f)$. Since $\mathcal{T}^{min}_{M}(f) = \mathcal{T}^{min}_{M^\psi}(f)$, we have to show $\mathcal{T}^{min}_{M^\psi}(f) \subseteq \mathcal{T}_{M^\psi}(f)$. However, this follows immediately from Definition 4.9.

- $\supseteq$ To show $\mathcal{G}^{min}_{M_\succsim}(\psi) \supseteq \mathcal{G}_{M^\psi_\succsim}(\psi)$, we have to show that $\mathcal{T}^{min}_{M}(f) \supseteq \mathcal{T}_{M^\psi}(f)$. Suppose this is not the case, i.e. suppose that $\mathcal{T}_{M^\psi}(f) \notin \mathcal{T}^{min}_{M}(f)$. Since $\mathcal{T}^{min}_{M}(f) = \mathcal{T}^{min}_{M^\psi}(f)$, we have that $\mathcal{T}_{M^\psi}(f) \notin \mathcal{T}^{min}_{M^\psi}(f)$. Then it must hold that there exists a normative game tree $T \in \mathcal{T}_{M^\psi}(f)$ such that $T \notin \mathcal{T}^{min}_{M^\psi}(f)$, implying that there exists a non-terminal history $h \in T \backslash \hat{T}$ such that $h \in dom_{M^\psi}(f)$. However, because Act_{H_h} is empty due to the definition of M^ψ found in Definition 4.19, it can never be the case that h is non-terminal. Contradiction!

□

Thus, given a multi-agent system M and state-based norm ψ, this proposition tells us that we can perform analysis on a structurally updated multi-agent system M^ψ to acquire results about the minimal games, in stead of arbitrary-length games. The following theorem is the main result of this chapter, and shows how we can apply our proof system to mechanism design and implementation theory.

Theorem 4.5. *Given a multi-agent system M, a set of agent types Θ and a state-based norm $\psi \in \mathcal{L}_{prop}(\Pi)$. We have that ψ ...*

1. ***Partial:*** *... partially SPNE-implements sc in M iff for all $\succsim \in \Theta$ we have:*

$$\vdash_{M^\psi_\succsim} (\phi^{hard}_{H_0} \wedge sc(\succsim)) \text{ leads-to } \psi$$

2. ***Weak:*** *... weakly SPNE-implements sc in M iff for all* $\succsim \in \Theta$ *and all soft states* S *such that* $S \notin \mathcal{O}_{sc(\succsim)}$ *we have:*

$$\nvdash_{M^{\psi}_{\succsim}} (\phi^{hard}_{H_0} \wedge \phi^{soft}_{S}) \text{ leads-to } \psi$$

3. ***Full:*** *... (fully) SPNE-implements sc iff* ψ *both partially and weakly SPNE-implements sc in M. Equivalently, for all* $\succsim \in \Theta$ *and all soft states* S *for which* $S \notin \mathcal{O}_{sc(\succsim)}$ *we have:*

$$\vdash_{M^{\psi}_{\succsim}} (\phi^{hard}_{H_0} \wedge sc(\succsim)) \text{ leads-to } \psi, \text{ and,}$$
$$\nvdash_{M^{\psi}_{\succsim}} (\phi^{hard}_{H_0} \wedge \phi^{soft}_{S}) \text{ leads-to } \psi$$

Proof. We only have to prove partial and weak implementation, since full implementation is just the combination of these two criteria.

1. **Partial:** From left-to-right and from right-to-left:

$$\begin{aligned}
\forall \succsim \in \Theta: \quad & \vdash_{M^{\psi}_{\succsim}} (\phi^{\text{hard}}_{H_0} \wedge sc(\succsim)) \text{ leads-to } \psi \overset{1}{\Leftrightarrow} \\
\forall \succsim \in \Theta: \quad & \vDash_{M^{\psi}_{\succsim}} (\phi^{\text{hard}}_{H_0} \wedge sc(\succsim)) \text{ leads-to } \psi \overset{2}{\Leftrightarrow} \\
\forall \succsim \in \Theta, \forall S \in \mathcal{O}_{sc(\succsim)}, \exists G \in \mathcal{G}^{min}_{M_{\succsim}}(\psi): \quad & S \in \mathcal{O}_{\text{SPNE}}(G) \overset{3}{\Leftrightarrow} \\
\forall \succsim \in \Theta, \exists G \in \mathcal{G}^{min}_{M_{\succsim}}(\psi), \forall S \in \mathcal{O}_{sc(\succsim)}: \quad & S \in \mathcal{O}_{\text{SPNE}}(G) \overset{4}{\Leftrightarrow} \\
\forall \succsim \in \Theta, \exists G \in \mathcal{G}^{min}_{M_{\succsim}}(\psi): \quad & \mathcal{O}_{sc(\succsim)} \subseteq \mathcal{O}_{\text{SPNE}}(G) \overset{5}{\Leftrightarrow} \\
& \psi \text{ partially SPNE-implements } sc \text{ in } M
\end{aligned}$$

The individual steps are explained below.

(1) Follows from Theorem 4.2 (soundness and completeness of system $\mathbf{ENS}(\mathbf{M}^{\psi}_{\succsim})$).

(2) Follows from Definition 4.17 and from Proposition 4.8. Particularly, given an arbitrary $\succsim \in \Theta$, we have $\vDash_{M^{\psi}_{\succsim}} (\phi^{\text{hard}}_{H_0} \wedge sc(\succsim))$ leads-to ψ if and only if for all H' and S' for which $H' \cup S' \vDash_{\text{prop}} (\phi^{\text{hard}}_{H_0} \wedge sc(\succsim))$, there exists a game $G \in \mathcal{G}_{(M^{\psi}, H')_{\succsim}}(\psi)$ such that $S' \in \mathcal{O}_{\text{SPNE}}(G)$. But since $\{(H_0, S) \mid S \in \mathcal{O}_{sc(\succsim)}\}$ are the only hard-soft state pairs that satisfy $(\phi^{\text{hard}}_{H_0} \wedge sc(\succsim))$, and since $\mathcal{G}_{(M^{\psi}, H_0)_{\succsim}}(\psi) = \mathcal{G}_{M^{\psi}_{\succsim}}(\psi) = \mathcal{G}^{min}_{M_{\succsim}}(\psi)$ (due to Proposition 4.8), the above reduces to $\forall S \in \mathcal{O}_{sc(\succsim)}$ there exists a game $G \in \mathcal{G}^{min}_{M_{\succsim}}(\psi)$ such that $S \in \mathcal{O}_{\text{SPNE}}(G)$.

(3) Follows from Proposition 4.5. Particularly, the set $\mathcal{G}^{min}_{M_{\succsim}}(\psi)$ is either empty or a singleton, so we can move the existential quantifier freely in or out of the scope of a universal quantifier.

(4) Definition of the subset relation '$\subseteq$'.

(5) Definition of partial implementation found in Definition 4.14.

2. **Weak:** From left-to-right and from right-to-left:

$$\forall \succsim \in \Theta, \forall S \notin \mathcal{O}_{sc(\succsim)}: \quad \nvdash_{M^{\psi}_{\succsim}} (\phi^{\text{hard}}_{H_0} \wedge \phi^{\text{soft}}_{S}) \text{ leads-to } \psi \overset{1}{\Leftrightarrow}$$

$$\forall \succsim \in \Theta, \forall S \notin \mathcal{O}_{sc(\succsim)}: \quad \nvdash_{M^{\psi}_{\succsim}} (\phi^{\text{hard}}_{H_0} \wedge \phi^{\text{soft}}_{S}) \text{ leads-to } \psi \overset{2}{\Leftrightarrow}$$

$$\forall \succsim \in \Theta, \forall S \notin \mathcal{O}_{sc(\succsim)}, \neg\exists G \in \mathcal{G}^{min}_{M_{\succsim}}(\psi): \quad S \in \mathcal{O}_{\text{SPNE}}(G) \overset{3}{\Leftrightarrow}$$

$$\forall \succsim \in \Theta, \forall S \notin \mathcal{O}_{sc(\succsim)}, \forall G \in \mathcal{G}^{min}_{M_{\succsim}}(\psi): \quad S \notin \mathcal{O}_{\text{SPNE}}(G) \overset{4}{\Leftrightarrow}$$

$$\forall \succsim \in \Theta, \forall G \in \mathcal{G}^{min}_{M_{\succsim}}(\psi), \forall S \notin \mathcal{O}_{sc(\succsim)}: \quad S \notin \mathcal{O}_{\text{SPNE}}(G) \overset{5}{\Leftrightarrow}$$

$$\forall \succsim \in \Theta, \forall G \in \mathcal{G}^{min}_{M_{\succsim}}(\psi): \quad \mathcal{O}_{\text{SPNE}}(G) \subseteq \mathcal{O}_{sc(\succsim)} \overset{6}{\Leftrightarrow}$$

$$\psi \text{ weakly SPNE-implements } sc \text{ in } M$$

The individual steps are again explained below.

(1) Follows from Theorem 4.2 (soundness and completeness of system $\mathbf{ENS}(\mathbf{M}^{\psi}_{\succsim})$).

(2) Follows from Definition 4.17 and from Proposition 4.8. Particularly, given an arbitrary $\succsim \in \Theta$ and $S \notin \mathcal{O}_{sc(\succsim)}$, we have $\nvdash_{M^{\psi}_{\succsim}} (\phi^{\text{hard}}_{H_0} \wedge \phi^{\text{soft}}_{S})$ leads-to ψ if and only if for all H' and S' for which $H' \cup S' \vDash_{\text{prop}} (\phi^{\text{hard}}_{H_0} \wedge \phi^{\text{soft}}_{S})$, there does not exist a game $G \in \mathcal{G}_{(M^{\psi}, H')_{\succsim}}(\psi)$ such that $S' \in \mathcal{O}_{\text{SPNE}}(G)$. But since (H_0, S) is the only hard-soft state pairs that satisfies $(\phi^{\text{hard}}_{H_0} \wedge \phi^{\text{soft}}_{S})$, and since $\mathcal{G}_{(M^{\psi}, H_0)_{\succsim}}(\psi) = \mathcal{G}_{M^{\psi}_{\succsim}}(\psi) = \mathcal{G}^{min}_{M_{\succsim}}(\psi)$ (due to Proposition 4.8), the above reduces to $\forall S \notin \mathcal{O}_{sc(\succsim)}$ there does not exist a game $G \in \mathcal{G}^{min}_{M_{\succsim}}(\psi)$ such that $S \in \mathcal{O}_{\text{SPNE}}(G)$.

(3) Moving of the logical negation in or out of the scope of the existential quantifier.

(4) Rearranging the universal quantifiers.

(5) Definition of the subset relation '$\subseteq$'.

(6) Definition of weak implementation found in Definition 4.14.

□

This concludes the main result of this chapter. From Theorem 4.4 we can thus conclude that determining (partial, weak, full) implementation in this framework is decidable. We now have the necessary tools to verify solutions of Solomon's dilemma with the use of our developed proof system. We will do this in the next section.

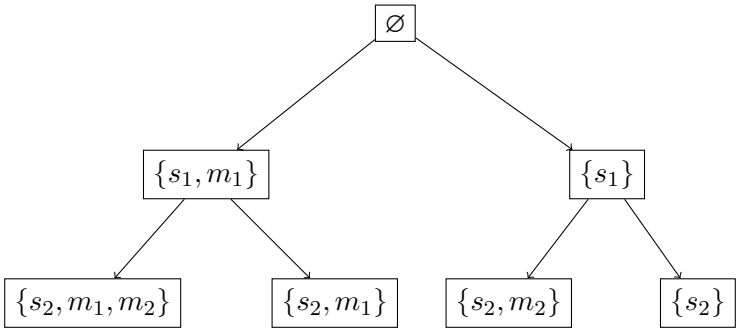


Figure 4.7: Multi-agent system M^{ψ}, in which we have removed all out-going actions of states which are in the domain of state-based norm ψ.

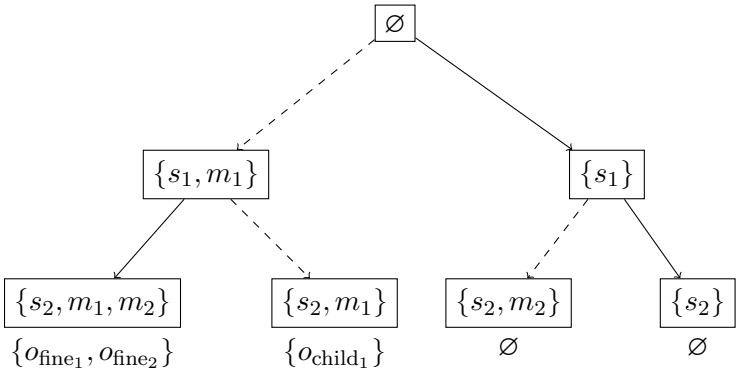
4.5.1 Solomon's dilemma solved

We return to the example of Solomon's dilemma, as presented in Section 4.3.1. The exact specifications of the system (i.e. the available actions and agents) can be found there, we only concern ourselves with how the dilemma can be solved. The solution, as proposed in [Moore, 1992], is as follows. Solomon decides the following: *"After both of you have spoken, if mother 1 claims not to be the real mother, the child goes to mother 2. If mother 1 claims to be the real mother and mother 2 does not, then the child goes to mother 1. If both of you claim to be the real mother then one of you is clearly lying. In this case the child goes to mother 2, but mother 1 should pay a small fine while mother 2 pays a big fine."*

Implicitly Solomon introduces the following state-based norm ψ to the multi-agent system:

$$\begin{aligned}\psi = ((&\phi^{\text{hard}}_{\{s_2,m_1,m_2\}} \wedge \phi^{\text{soft}}_{\{o_{\text{fine}_1},o_{\text{fine}_2}\}}) \vee\\ (&\phi^{\text{hard}}_{\{s_2,m_1\}} \wedge \phi^{\text{soft}}_{\{o_{\text{child}_1}\}}) \vee\\ (&\phi^{\text{hard}}_{\{s_2,m_2\}} \wedge \phi^{\text{soft}}_{\emptyset}) \vee\\ (&\phi^{\text{hard}}_{\{s_2\}} \wedge \phi^{\text{soft}}_{\emptyset}))\end{aligned}$$

That is, whenever s_2 is true (mother 1 and mother 2 have both spoken), if it is the case that m_1 and m_2 are true (mother 1 and mother 2 both claim to be the real mother), they both receive a fine (o_{fine_1} and o_{fine_2}) and mother 2 receives the child ($\neg o_{\text{child}_1}$). If m_1 is true and m_2 is false (only mother 1 claims to be the real mother), then mother 1 receives the child (o_{child_1}) without any fine given ($\neg o_{\text{fine}_1}$ and $\neg o_{\text{fine}_2}$). If m_1 is false (mother 1 claims not to be the real mother), then mother 2 receives the child ($\neg o_{\text{child}_1}$) without any fines



$$\dfrac{\dfrac{\dfrac{\{(\phi^{\text{hard}}_{\varnothing} \wedge \phi^{\text{soft}}_{\{o_{\text{child}_1}\}}) \wedge \neg\psi'\} M^{\psi}_{\gtrsim} \{\psi'\}}{(\phi^{\text{hard}}_{\varnothing} \wedge \phi^{\text{soft}}_{\{o_{\text{child}_1}\}}) \text{ ensures } \psi'}\ (\text{ENS}) \qquad \dfrac{\{\psi' \wedge \neg\psi\} M^{\psi}_{\gtrsim} \{\psi\}}{\psi' \text{ ensures } \psi}\ (\text{ENS})}{(\phi^{\text{hard}}_{\varnothing} \wedge \phi^{\text{soft}}_{\{o_{\text{child}_1}\}}) \text{ leads-to } \psi'}\ (\text{LT1}) \qquad \dfrac{}{\psi' \text{ leads-to } \psi}\ (\text{LT1})}{(\phi^{\text{hard}}_{\varnothing} \wedge \phi^{\text{soft}}_{\{o_{\text{child}_1}\}}) \text{ leads-to } \psi}\ (\text{LT2})$$

$$\text{Where } \psi' = (\phi^{\text{hard}}_{\{s_1, m_1\}} \wedge sc(\gtrsim)) \vee (\phi^{\text{hard}}_{\{s_1\}} \wedge sc(\gtrsim'))$$

Figure 4.8: Deduction for $(\phi^{hard}_{H_0} \wedge \phi^{soft}_{\{o_{\text{child}_1}\}})$ *leads-to* ψ*. On the top, we have drawn the corresponding game, where the dashed edges show the SPNE strategy profile*

given ($\neg o_{\text{fine}_1}$ and $\neg o_{\text{fine}_2}$). The updated multi-agent system M^{ψ} as defined in Definition 4.19 is drawn in Figure 4.7.

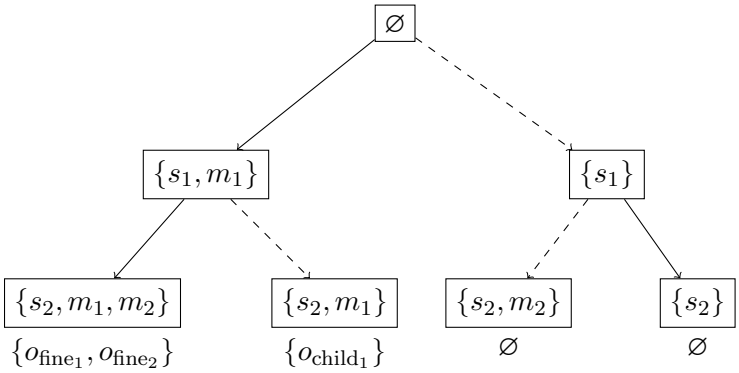
We now claim that ψ fully implements sc in M, and we can use our proof system to show that this is the case. We will focus our attention on partial implementation. We invoke Theorem 4.5, which tells us that in order to show that a state-based norm ψ partially implements sc in M, we have to show that for all $\gtrsim \in \Theta$ we have:

$$\vdash_{M^{\psi}_{\gtrsim}} (\phi^{\text{hard}}_{H_0} \wedge sc(\gtrsim)) \text{ leads-to } \psi$$

That is, we have to show for all $\gtrsim \in \Theta$ that:

$$\vdash_{M^{\psi}_{\gtrsim}} (\phi^{\text{hard}}_{H_0} \wedge \phi^{\text{soft}}_{\{o_{\text{child}_1}\}}) \text{ leads-to } \psi, \text{ and,}$$
$$\vdash_{M^{\psi}_{\gtrsim'}} (\phi^{\text{hard}}_{H_0} \wedge \phi^{\text{soft}}_{\varnothing}) \text{ leads-to } \psi$$

We recall from Section 4.3.1 that $H_0 = \varnothing$. We show them separately:



$$\dfrac{\dfrac{\dfrac{\{(\phi^{\text{hard}}_{\varnothing} \wedge \phi^{\text{soft}}_{\varnothing}) \wedge \neg\psi'\} M^{\psi}_{\succsim'} \{\psi'\}}{(\phi^{\text{hard}}_{\varnothing} \wedge \phi^{\text{soft}}_{\varnothing}) \text{ ensures } \psi'}\ (\text{ENS})}{(\phi^{\text{hard}}_{\varnothing} \wedge \phi^{\text{soft}}_{\varnothing}) \text{ leads-to } \psi'}\ (\text{LT1}) \qquad \dfrac{\dfrac{\{\psi' \wedge \neg\psi\} M^{\psi}_{\succsim'} \{\psi\}}{\psi' \text{ ensures } \psi}\ (\text{ENS})}{\psi' \text{ leads-to } \psi}\ (\text{LT1})}{(\phi^{\text{hard}}_{\varnothing} \wedge \phi^{\text{soft}}_{\varnothing}) \text{ leads-to } \psi}\ (\text{LT2})$$

$$\text{Where } \psi' = (\phi^{\text{hard}}_{\{s_1,m_1\}} \wedge sc(\succsim)) \vee (\phi^{\text{hard}}_{\{s_1\}} \wedge sc(\succsim'))$$

Figure 4.9: Deduction for $\vdash_{M^{\psi}_{\succsim}} (\phi^{hard}_{H_0} \wedge \phi^{soft}_{\varnothing})$ *leads-to* ψ*. On the top, we have drawn the corresponding game, where the dashed edges show the SPNE strategy profile.*

1. By applying rules and axioms from system $\mathbf{LT}(\mathbf{M}^{\psi}_{\succsim})$, in Figure 4.8 we have shown the inference tree for the assertion:

   $$\vdash_{M^{\psi}_{\succsim}} (\phi^{\text{hard}}_{\varnothing} \wedge \phi^{\text{soft}}_{\{o_{\text{child}_1}\}}) \text{ leads-to } \psi$$

   On the top, we have drawn the corresponding game, where the dashed edges show the Sub-game Perfect Nash Equilibrium strategy profile and at the bottom we have shown the corresponding outcomes. This visualises that it is indeed the case that outcome $sc(\succsim) = \{o_{\text{child}_1}\}$ is achieved. The validity of $\{(\phi^{\text{hard}}_{\varnothing} \wedge \phi^{\text{soft}}_{\{o_{\text{child}_1}\}}) \wedge \neg\psi'\} M^{\psi}_{\succsim} \{\psi'\}$ and $\{\psi' \wedge \neg\psi\} M^{\psi}_{\succsim} \{\psi\}$ (where $\psi' = (\phi^{\text{hard}}_{\{s_1,m_1\}} \wedge sc(\succsim)) \vee (\phi^{\text{hard}}_{\{s_1\}} \wedge sc(\succsim'))$) needs to be shown separately by using Definition 4.15 or by using the interpretation found in Lemma 4.1.

2. By applying rules and axioms from system $\mathbf{LT}(\mathbf{M}^{\psi}_{\succsim'})$, in Figure 4.9 we have shown the inference tree for the assertion:

   $$\vdash_{M^{\psi}_{\succsim'}} (\phi^{\text{hard}}_{\varnothing} \wedge \phi^{\text{soft}}_{\varnothing}) \text{ leads-to } \psi$$

On the top, we have again drawn the corresponding game, where the dashed edges show the Sub-game Perfect Nash Equilibrium strategy profile and at the bottom we have shown the corresponding outcomes. This again visualises that it is indeed the case that outcome $sc(\succsim') = \varnothing$ is achieved. The validity of $\{(\phi^{\text{hard}}_{\varnothing} \wedge \phi^{\text{soft}}_{\varnothing}) \wedge \neg\psi'\}M^{\psi}_{\succsim'}\{\psi'\}$ and $\{\{\psi' \wedge \neg\psi\}M^{\psi}_{\succsim'}\{\psi\}$ (where $\psi' = (\phi^{\text{hard}}_{\{s_1,m_1\}} \wedge sc(\succsim)) \vee (\phi^{\text{hard}}_{\{s_1\}} \wedge sc(\succsim')))$ needs to be shown separately by using Definition 4.15 or by using the interpretation found in Lemma 4.1.

This concludes our example of Solomon's dilemma. Note that in order to prove full implementation, we need to prove weak implementation as well, which is left to the reader.

4.6 Discussion

In this chapter, inspired by the proof system of Unity, we developed a logical proof system that allows us to specify and verify various implementability questions related to normative multi-agent systems. Particularly, given the specification of a multi-agent system, a set of possible agent types, a social choice rule denoting what is optimal for society, and a sequence-based norm acting as the underling mechanism, this proof system can be used to verify whether this mechanism implements the social choice rule in sub-game perfect equilibrium.

This chapter can be extended in multiple ways, which we will briefly reflect upon. It is interesting to consider different mechanisms (e.g. other classes of norms) and different solution concepts (e.g. non-sub-game perfect Nash Equilibria). Moreover, we assume that the agents in this system possess complete information of the multi-agent system, which is perhaps a strong assumption that can be weakened, leading to the field of *epistemic game theory*. Moreover, we related our implementability questions to minimal normative games, but it might be interesting to consider other kinds of normative games that can occur. Finally, it is of importance to study the social choice theory questions surrounding normative mechanism design. Particularly, it is interesting to ask what suitable criteria are for optimal outcomes. For example, a typical criterion one may adopt in a utilitarian society is that wrong-doers should be punished. On the other hand, punishment has consequences for both the offender and society. Thus, another criterion might be that somebody should only be punished if the total good produced by the punishment should exceed the total evil. In this chapter, we assumed that the social choice rule was already pre-determined.

Chapter 5

Modelling and Verifying Dynamic Systems

In the previous chapters, we saw that a norm, when added to a multi-agent system, was able to change the behaviour of such a system. When norms are added to a system, it may for example be that certain actions become forbidden, or certain states become allowed. In this chapter, we concern ourselves with how to model and verify the dynamics of a normative multi-agent system under addition of various classes of norms in order to prove correctness of these systems.

5.1 Introduction

In this chapter, our goal is to reason about, and prove properties of, dynamic normative multi-agent systems. A dynamic normative multi-agent system is a multi-agent system governed by norms in which these norms may change over time (that is, during execution of these system). One may identify two possible methodological approaches when dealing with norm change. On the one hand we can identify the syntactic approach, as found in [Boella et al., 2009], where norm change is considered as an operation on the underlying "code" that constitutes the system. On the other hand we can identify the semantic approach as introduced in our earlier work in [Knobbout et al., 2014], which tries to look at norm change as an update of the model. However, in this work we did not consider a proof system to reason about these updates in order to perform formal verification. Related to the semantic approach, in [Aucher et al., 2009] the authors give a logical account of ought-to-be norm change via the notions of context expansion and contraction. In our work, we perform norm updates on pointed labelled transition systems to give a logical account of a much more general form of norm change via updates of the whole

system at hand.

The view we adopt in this chapter is that (normative) multi-agent systems can be modelled by pointed labelled transition systems, which show which (normative) facts become true under execution of which actions. Adding a norm is an operation on these models: they transform normative multi-agent systems such that the resulting system is aligned with the added norm. In this chapter we explore how these operations work, and how we can use these operations in combination with our logic to prove various properties of a normative multi-agent system in a dynamic environment. This work is inspired by the work of Dynamic Epistemic Logic (DEL) in the sense that we also explore how we can reason about dynamic updates of a given transition system, see [van Ditmarsch et al., 2007]. However, we consider the setting of normative multi-agent systems, where the structures are normative multi-agent systems instead of epistemic models, and the updates are norms instead of epistemic formulas.

5.1.1 The verification problem

The verification problem we consider in this chapter is the following: Given a specific behaviour of a normative multi-agent system in a dynamic environment, where the behaviour consists of actions that have taken place and normative updates that can occur, verify whether the desired effects are satisfied. The corresponding synthesis and design question is whether we can synthesize a normative system in order to ensure that for specific behaviours the desired effects are satisfied.

5.1.2 Chapter outline

We give a brief outline of the contents of this chapter:

- In **Section 5.2** (p 111) we introduce the model of execution we are interested in, define a basic modal language to reason about the execution of actions, and end with an example to show how we can use this logical language in order to reason about properties of normative systems.
- In **Section 5.3** (p 115) we explore norms of the 'to-be' variant. These kinds of norms can for example forbid (or allow) certain states to occur, and thus modify the behaviour of a normative multi-agent system. We provide update semantics, create a logical language that allows us to reason about these updates, and provide a sound and complete proof system of this logic.
- In **Section 5.4** (p 131) we move to norms of the 'to-do' variant, which may for example forbid certain actions. We again provide update se-

mantics, create an extended logical language to reason about both these updates and the updates of the previous section, and give an extended proof system which is again both sound and complete.

- In **Section 5.5** (p 150) we discuss this chapter.

5.2 Preliminaries

In this section we define the models we use for normative multi-agent systems, and present the syntax and semantics of the logic we use to express properties of these systems.

5.2.1 Normative labelled transition systems

In the previous chapter, we saw that (enforcement) norms could cause certain positive or negative sanctions to be in effect if the behaviour leading up to such a state was considered good or bad behaviour. In this chapter we want to reason about how norms can dynamically change the behaviour of a system. In order to do this, we view a normative multi-agent system as simply a multi-agent system (containing agents, actions and propositions), but with the addition of soft facts representing the positive and negative sanctions. This system then tells us for each possible execution of actions the soft and hard facts which become true. In this chapter, we do not wish to model which agent (or group of agents) is responsible for these actions, we merely want to reason about the *effect* of such actions. Because of this simplification, we do not have to represent the agents explicitly, and can just assume the existence of some action alphabet. Formally, we assume we have such an alphabet Act which denotes the set of possible actions that can occur. When a certain action $\alpha \in Act$ occurs, we progress from a state to another state in which certain soft and hard facts become true. We represent such a model as a *normative labelled transition system*.

Definition 5.1 (Normative labelled transition systems)**.** *A* normative labelled transition system N *is a tuple* $(Q, \Pi, \mu, Act, \rightarrow)$ *such that:*

- Q *is a non-empty finite set of states.*
- $\Pi = \Pi_{\text{hard}} \cup \Pi_{\text{soft}}$ *is a finite set of propositions such that* $\Pi_{\text{hard}} \cap \Pi_{\text{soft}} = \varnothing$. *The set* Π_{hard} *is referred to as the set of hard facts, and* Π_{soft} *as the set of soft facts.*
- $\mu : Q \rightarrow \mathcal{P}(\Pi)$ *is a valuation function mapping a state* $q \in Q$ *to an element from* $\mathcal{P}(\Pi)$.

- *Act is a finite set of (domain) actions.*
- $\rightarrow \subseteq Q \times Act \times Q$ *is a relation between states with actions, such that for all* $q \in Q$ *and* $\alpha \in Act$ *there exists exactly one* $q' \in Q$ *such that* $(q, \alpha, q') \in \rightarrow$ *(the relation is functional). Whenever* $(q, \alpha, q') \in \rightarrow$*, we write* $q(\alpha)$ *to denote* q'*, and we write* $q(\alpha_1 \dots \alpha_n)$ *to denote* $((q(\alpha_1))(\dots))(\alpha_n)$*.*

A pointed normative labelled transition system is a pair (N, q) *such that* N *is a normative system, and* $q \in Q$ *a state from* N*.*

In the remainder of this chapter, we refer to such a system simply as a (pointed) normative system instead of a (pointed) labelled transition system. A normative system differs from a "standard" labelled transition system in the following two important ways:

1. We explicitly distinguish between hard and soft facts. Similarly as in Chapter 4, norms can add or remove soft facts in the system. This is a topic which we will discuss later.
2. We assume that every action is always applicable, and that the resulting state of performing such an action is decided in a deterministic manner. Whenever we want to model that an action has no effect in a certain state, we can simply model this with a reflexive transition.

Relating to the various properties of multi-agent systems we discussed in Chapter 2, this model is thus synchronous, decentralized, discrete and deterministic. In the next section we will introduce a modal logic to reason about the effects of actions (in terms of hard and soft facts which become true or false) in states.

5.2.2 Modal action logic

In this section, we will introduce a basic language to reason about the effects of actions in states. This allows us to determine what soft and hard facts become true after the performance of certain actions. Later, we will see how we can *extend* this language to also take into account the addition of norms. Given a normative system $N = (Q, \Pi, \mu, Act, \rightarrow)$, the language of propositional logic with action modality, written in this chapter as $\mathcal{L}_0$, consists of formulas φ built by the following grammar, where $p \in \Pi$ and $\alpha \in Act$:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \Box_\alpha \varphi$$

Along a pointed normative system (N, q), we can evaluate formulas of $\mathcal{L}_0$ in the following way.

Definition 5.2 ($\mathcal{L}_0$ semantics)**.** *Given a normative system* $N = (Q, \Pi, \mu, Act, \rightarrow)$ *and a state* $q \in Q$*, the satisfaction relation* $\models_{L_0}$ *of* $\mathcal{L}_0$ *is inductively defined as follows:*

- $N, q \vDash_{L_0} p$ *iff* $p \in \mu(q)$
- $N, q \vDash_{L_0} \neg\varphi$ *iff* $N, q \nvDash_{L_0} \varphi$
- $N, q \vDash_{L_0} \varphi_1 \vee \varphi_2$ *iff* $N, q \vDash_{L_0} \varphi_1$ *or* $N, q \vDash_{L_0} \varphi_2$.
- $N, q \vDash_{L_0} \Box_\alpha \varphi$ *iff* $N, q(\alpha) \vDash_{L_0} \varphi$

We write $\vDash$ instead of $\vDash_{L_0}$ whenever clear from context. Given a pointed normative system (N, q), we say that a sequence of actions $\alpha_1 \dots \alpha_n$ *brings about* state of affairs φ if and only if $N, q \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} \varphi$. As is standard, we say that $N \vDash \varphi$ holds whenever for all $q \in Q$ it holds that $N, q \vDash \varphi$, and we say that $\vDash \varphi$ holds (alternatively, "φ is valid") whenever for all normative systems N we have $N \vDash \varphi$. We have the following result from standard theory on modal logic.

Theorem 5.1 (Sound and complete axiomatization of $\mathcal{L}_0$)**.** *The logic $\mathcal{L}_0$ is soundly and completely axiomatized by system* **K** *(i.e. the system consisting of all propositional tautologies together with the necessitation rule and the distributive axiom) with the following 'Function' axiom scheme:*

$$\textit{(Function)} \quad \Box_\alpha \neg\varphi \leftrightarrow \neg \Box_\alpha \varphi$$

We refer to system **K** *with this additional axiom scheme as system* **L0***, and write $\vdash_{L_0} \varphi$ whenever φ is provable in* **L0***. For all $\varphi \in \mathcal{L}_0$ we thus have:*

$$\vDash_{L_0} \varphi \quad \Leftrightarrow \quad \vdash_{L_0} \varphi$$

For a full treatment of system **K** and several modal axioms, we direct the reader to [Chellas, 1980]. The additional axiom schemata in $\mathcal{L}_0$ states that the arrows of the normative system are functional, i.e. a state and an action completely and uniquely define the next state. For now, we will briefly discuss an example normative system of a library which lends out books to customers. This will show how we can evaluate formulas from our logic on pointed normative system. This example will also function as a running example throughout this chapter to demonstrate how various norms may alter the behaviour of the system.

5.2.3 Running example

Consider the normative system N_{library} in Figure 5.1. This simple system consists of a library agent and one costumer, and we are interested in the interaction that arises between them. The action 'lend_book' is considered an action of the library, and the action 'return' an action of the agent, although as mentioned earlier we are not interested in explicitly modelling which agent

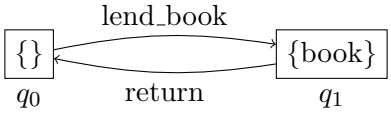


Figure 5.1: System $N_{library}$ consisting of two states q_0 and q_1.

is responsible for which action. Here '*book*' denotes the fact that the customer has a book in its possession. The system can either be in state q_0 or state q_1, and depending on the performed actions can alter between these states. It is important to note that we omit drawing the reflexive arrows; i.e. action 'return' performed in state q_1 will result in q_1. An example of a formula that holds in this system is:

$$N_{\text{library}}, q_1 \vDash \mathit{book} \wedge \Box_{\text{return}}(\neg \mathit{book})$$

Later on we will add norms to this system and see how the system can show more interesting and complex behaviour.

5.2.4 Norm classes

Before we formally introduce the language of norms and norm update, it is worthwhile to reflect on the classes of norms we want to express and model. Broadly speaking, we consider the following two classes:

1. **State-based norms:** A state-based norm refers to certain states that should be achieved or avoided. These kinds of norms are of the 'to-be' variant. For example, the library might add the norm which states that having a book is forbidden. To extend the expressiveness of these kinds of norms even further, we also optionally allow the addition of a repair action. For example, the library may state that having a book is forbidden, until a subscription is bought. These state-based norms are thus conditional on a repair action, and this approach is to our knowledge new and allows for very expressive norms. This language will be formally introduced in Section 5.3.

2. **Action-based norms:** An action-based norm refers to certain actions that should be performed or avoided. These kinds of norms are of the 'to-do' variant. For example, the library might add the norm which states that lending a magazine is forbidden. Again, we optionally allow the performance of a repair actions, which allows for expressive norms. This language will be formally introduced in Section 5.4.

Moreover, more elaborate norms pertaining to complex behaviours of the system, like process-based norms, can be acquired by combining norms in various ways. Process-based norms refers to certain sequences of actions that should be performed of avoided. For example, the library might add the norm that states that lending a magazine should always be preceded by a payment action. It is important to note that the addition of these norms are not 'physical' actions of the system. They come from outside the system (i.e. a designer) and are not triggered by the domain actions.

5.3 Language for norm update

In this section we begin our first step into developing a simple model and language of normative update that is able to capture a broad variety of different kinds of norms. The kinds of norms we consider in this section are of the 'to-be' variant, and correspond to the state-based norms we mentioned in Section 5.2.4. A norm of this class, when added to the system, causes that in certain states certain soft facts become true or false. To immediately relate it to our previous example consisting of the library and the agent, a norm might be added which states that having a book is forbidden (i.e., for all '*book*' states violation 'v' will start to hold). To extend the expressiveness of these kinds of norms even further, we also optionally allow the addition of a repair action. For example, the library may state that having a book is forbidden, until a subscription is bought (from which on, having a book is no longer forbidden).

In the remainder of this section, we give a language to construct these kinds of norms, we show how we can update a normative system using these norms and finally we show how we can add these constructs to our logical language.

5.3.1 Language and update

Given a normative system $N = (Q, \Pi, \mu, Act, \rightarrow)$, we construct the norm language $\mathcal{N}_1$ in the following way, where $\varphi \in \mathcal{L}_0$, $p \in \Pi_{\text{soft}}$ and $Act_R \subseteq Act$:

$$\mathsf{n} ::= (\varphi, +p, Act_R) \mid (\varphi, -p, Act_R)$$

Whenever we have a norm $(\varphi, \pm p, Act_R)$ (where $\pm p$ can either be $+p$ or $-p$), we refer to φ as the norm condition, $\pm p$ as the norm effect and Act_R as the repair actions. These constructs relate to the kinds of norms we described earlier. Adding a norm $(\varphi, +p, Act_R)$ implies that for every φ-state the soft fact p will start to hold, until a repair action from the set Act_R occurs. Whenever a repair action occurs, we say that the norm effect is repaired. Note that we use the notion of repair in a rather liberal way, since a repair action may

very well occur before a φ-state is ever encountered. In this case, we still say that a norm effect is repaired, even though it was never the case that we were in a state in which the norm effect $\pm p$ was realized. This is especially true for a norm of the form $(\bot, \pm p, Act_R)$, since no state ever satisfies $\bot$, but we would still say that the effect is repaired whenever an action from Act_R occurs. As we will see later, because the norm effect of this particular norm is never realized, this norm does not change the behaviour of a normative system. We additionally note that the set Act_R can be empty, which implies that the update is permanent. For example, the norm $(\varphi, +v, \varnothing)$ states that all φ-states are now permanently forbidden, causing violation v to be in effect. In this way, we acquire the 'classical' interpretation of a state-based norm, which is related to the idea of Anderson's reduction [Anderson, 1958]. The norm $(\varphi, -p, Act_R)$ behaves in a similar fashion, except in this case the soft fact p will stop to hold until this norm effect is repaired. These updates work on pointed normative systems. That is, they transform a pointed normative system to a new pointed normative system. In this chapter we want to give a clear semantic interpretation to how a pointed normative system should behave when a norm from our language is added to the system. This is why we introduce the notion of *norm-aligned*. We say that an updated system is norm-aligned if the resulting system implements the new restrictions of the added norm.

Postulate 5.1 (Norm-Aligned updates from $\mathcal{N}_1$)**.** *Any normative system updated with a norm* $\mathsf{n} \in \mathcal{N}_1$ *should be norm-aligned with* n*. Given a pointed normative system* (N, q) *and* $\mathsf{n} \in \mathcal{N}_1$*, let* $(N, q)'$ *be the system updated with* n*. We distinguish between the cases where* $\mathsf{n} = (\varphi, +p, Act_R)$ *or where* $\mathsf{n} = (\varphi, -p, Act_R)$*. We say that* $(N, q)'$ *is* norm-aligned *with ...*

1. *...* $(\varphi, +p, Act_R)$ *iff for every atomic proposition* $p' \in \Pi$ *and every (possibly empty) finite sequence of actions* $\alpha_1 \dots \alpha_n \in Act^*$*, we have that* $(N, q)' \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p'$ *if and only if:*

   (a) $N, q \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p'$, ***or;***

   (b) $p = p'$, $N, q \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} \varphi$, *and,* $\alpha_1, \dots, \alpha_n \notin Act_R$.

2. *...* $(\varphi, -p, Act_R)$ *iff for every atomic proposition* $p' \in \Pi$ *and every (possibly empty) finite sequence of actions* $\alpha_1 \dots \alpha_n \in Act^*$*, we have that* $(N, q)' \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p'$ *if and only if:*

   (a) $N, q \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p'$, ***and;***

   (b) $p \neq p'$, *or,* $N, q \nvDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} \varphi$, *or,* $\exists \alpha_i \, (1 \leq i \leq n) : \alpha_i \in Act_R$.

In words, when we update with a norm $(\varphi, +p, Act_R)$, action $\alpha_1 \dots \alpha_n$ should bring about p' in the updated system iff either (1) $\alpha_1 \dots \alpha_n$ already

brought about p' in the original system before the update ($N, q \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p'$), or (2) p' is added by the norm effect ($p' = p$), the norm condition is brought about by $\alpha_1 \ldots \alpha_n$ ($N, q \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} \varphi$), and the norm effect was not already repaired ($\alpha_1, \ldots, \alpha_n \notin Act_R$). This notion captures both a property of success and a property of minimal change. On the one hand, it states that the norm effect should hold under the right conditions (property of success corresponding to item 2), and on the other hand it states that everything else should remain unchanged (property of minimal change corresponding to item 1). This is inspired by the kind of postulates we can find in the AGM framework as discussed in [Alchourrón et al., 1985], which state how updates should behave. Although these norms may appear at first quite limited in expressiveness, more complex norms can be expressed by combining a multitude of these simpler norms.

We will now show how we can update a pointed normative system to a new pointed normative system which, as we will show later, will be norm-aligned. Given a pointed normative system (N, q) and a norm n, we write $(N, q)[\mathsf{n}]$ to denote the updated system.

Definition 5.3 (Updates from $\mathcal{N}_1$)**.** *Given $N = (Q, \Pi, \mu, Act, \rightarrow)$, $q \in Q$ and $\mathsf{n} = (\varphi, +p, Act_R) \in \mathcal{N}_1$, we let $(N, q)[\mathsf{n}] = (N[\mathsf{n}], q[\mathsf{n}])$ such that:*

- *$N[\mathsf{n}] = (Q', \Pi, \mu', Act, \rightarrow')$, where:*
  - *$Q' = \{q^r, q^a \mid q \in Q\}$*
  - *For every $q \in Q$:*

$$\mu'(q^r) = \mu(q) \quad \text{and} \quad \mu'(q^a) = \begin{cases} \mu(q) \cup \{p\} & \text{if } N, q \vDash \varphi \\ \mu(q) & \text{otherwise} \end{cases}$$

  - $$\rightarrow' = \begin{array}{l} \{(q_i^a, \alpha, q_j^a) \mid q_i(\alpha) = q_j \text{ and } \alpha \notin Act_R\} \cup \\ \{(q_i^a, \alpha, q_j^r) \mid q_i(\alpha) = q_j \text{ and } \alpha \in Act_R\} \cup \\ \{(q_i^r, \alpha, q_j^r) \mid q_i(\alpha) = q_j\} \end{array}$$

- *State $q[\mathsf{n}] = q^a$*

Note that we have written q_i and q_j in this definition to simply denote that these states might possibly be different; we attach no further meaning to this indexing. Alternatively, for an update with $-p$, the updated valuation function μ' becomes:

$$\mu'(q^r) = \mu(q) \text{ and } \mu'(q^a) = \begin{cases} \mu(q) \backslash \{p\} & \text{if } N, q \vDash \varphi \\ \mu(q) & \text{otherwise} \end{cases}$$

That is, for every state $q \in Q$ we create two copies in the updated system; one in which the norm effect is active (q^a) and one in which it is repaired (q^r). The

transitions between active and repaired states are analogous to the transitions of the original system, except whenever we are in an active state and a repair action occurs, in which case we go to a repaired state. These kind of updates are inspired by the kind of updates we may see in Dynamic Epistemic Logic (DEL) such as Public Announcement Logic (PAL) (see [van Ditmarsch et al., 2007]), but note that the argument (a norm, which is not just a logical formula but a complex structure consisting of a condition, effect and repair actions) and result (a norm-aligned system) of our update is something completely different than what we may find there. Lastly, as an important note, we want to express that although these constructs may at first glance be related to Propositional Dynamic Logic (PDL) [Fischer and Ladner, 1977], the updates we consider are performed on 'higher order' Kripke structures and not just on state-valuations. The following proposition shows that these updates result in a norm-aligned system. Note that we only show $(\varphi, +p, Act_R)$, since the proof for $(\varphi, -p, Act_R)$ is similar.

Proposition 5.1. *Given* $N = (Q, \Pi, \mu, Act, \rightarrow)$, $q \in Q$ *and* $\mathsf{n} = (\varphi, +p, Act_R) \in \mathcal{N}_1$, *the pointed normative system* $(N, q)[\mathsf{n}]$ *is norm-aligned with* n.

Proof. We assume an arbitrary finite (possibly empty) sequence of actions $\alpha_1 \dots \alpha_n \in Act^*$ and proposition p'. We have:

$$(N, q)[\mathsf{n}] \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p' \quad \Leftrightarrow \quad N[\mathsf{n}], q^a \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p'$$

We consider the case where $\alpha_1, \dots, \alpha_n \notin Act_R$ or the case where there exists a α_i $(1 \leq i \leq n)$ for which $\alpha_i \in Act_R$:

- If $\alpha_1, \dots, \alpha_n \notin Act_R$, we have:

$$\begin{gathered} N[\mathsf{n}], q^a \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p' \quad \Leftrightarrow \quad N[\mathsf{n}], q(\alpha_1 \dots \alpha_n)^a \vDash p' \quad \Leftrightarrow \\ N, q(\alpha_1 \dots \alpha_n) \vDash p' \text{ or } (p' = p \text{ and } N, q(\alpha_1 \dots \alpha_n) \vDash \varphi) \quad \Leftrightarrow \\ N, q \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p' \text{ or } (p' = p \text{ and } N, q \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} \varphi) \end{gathered}$$

- If there exists a α_i $(1 \leq i \leq n)$ for which $\alpha_i \in Act_R$, we have:

$$\begin{gathered} N[\mathsf{n}], q^a \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p' \quad \Leftrightarrow \quad N[\mathsf{n}], q(\alpha_1 \dots \alpha_n)^r \vDash p' \quad \Leftrightarrow \\ N, q(\alpha_1 \dots \alpha_n) \vDash p' \quad \Leftrightarrow \quad N, q \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p' \end{gathered}$$

□

With our running example of the library, we can visualize how this update changes the behaviour of a normative system.

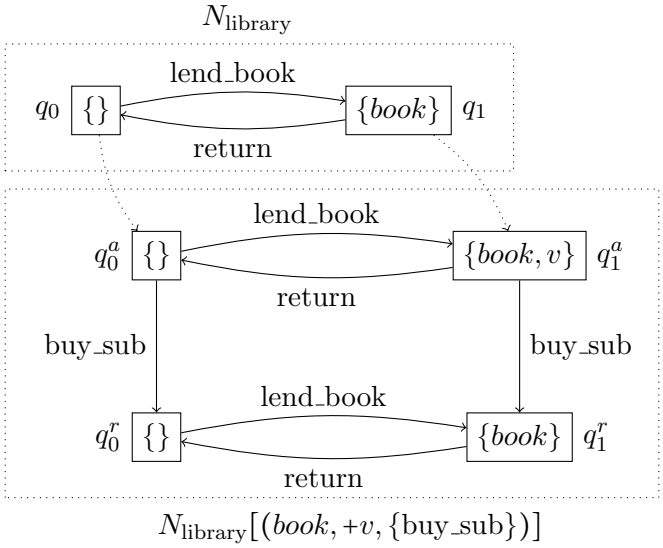


Figure 5.2: On top we have normative system $N_{library}$, and on the bottom we have normative system $N_{library}[(book, +v, \{buy\_sub\})]$. The dotted transitions denote state-updates that occur.

Running Example

We will return to the library normative system found in Figure 5.1. Assume that the library agent had the tradition to lend books freely, and that we now want to impose a policy such that the customer agent needs to buy a subscription to have a book in his possession, otherwise he is in violation. Formally, the system is updated with the norm n_0 = (*book*, $+v$, {buy_sub}), where 'buy_sub' is an action of the agent. This norm states that to be in a '*book*'-state causes violation v, unless this effect is repaired by buying a subscription. In Figure 5.2 we see how we update the normative system N_{library} to system $N_{\text{library}}[\mathsf{n}_0]$, and states q_0 and q_1 to q_0^a and q_1^a respectively. Note that the 'buy_sub' action was not drawn in Figure 5.1 due to the omission of the reflexive arrows. It is important to note that the update can affect the current state; if we would be in state q_1 in the original system, it could be that an update causes us to be a violation-state. That is:

$$(N_{\text{library}}, q_1)[\mathsf{n}_0] \vDash v$$

If the *lend* action is performed before the *buy_sub* action in the updated system, we would also be in a violation-state. That is:

$$(N_{\text{library}}, q_0)[\mathsf{n}_0] \vDash \Box_{\text{lend\_book}} v$$

Otherwise this would not be the case:

$$(N_{\text{library}}, q_0)[\mathsf{n}_0] \vDash \Box_{\text{buy\_sub}} \Box_{\text{lend\_book}} \neg v$$

Thus, we see that this relatively simple norm already makes the behaviour of the system more interesting and complex. In the next section we will extend our logical language to allow us to reason about these dynamic updates.

5.3.2 Syntax and semantics

In this section, we show how we can extend language $\mathcal{L}_0$ to allow us to reason about these norm updates. Given a normative system $N = (Q, \Pi, \mu, Act, \rightarrow)$, the language of propositional logic with action modality including norm updates from norm language $\mathcal{N}_1$, written in this chapter as $\mathcal{L}_1$, consists of formulas φ built by the following grammar, where $p \in \Pi$, $\alpha \in Act$, and $\mathsf{n} \in \mathcal{N}_1$:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \Box_\alpha \varphi \mid [\mathsf{n}]\varphi$$

From this grammar it is immediately apparent that $\mathcal{L}_0 \subset \mathcal{L}_1$. Along a pointed normative system (N, q), we can evaluate formulas of $\mathcal{L}_1$ as follows.

Definition 5.4 ($\mathcal{L}_1$ semantics)**.** *Given a normative system $N = (Q, \Pi, \mu, Act, \rightarrow)$ and a state $q \in Q$, the satisfaction relation $\vDash_{L_1}$ of $\mathcal{L}_1$ is inductively defined as follows:*

- $N, q \vDash_{L_1} p$ *iff* $p \in \mu(q)$
- $N, q \vDash_{L_1} \neg\varphi$ *iff* $N, q \nvDash_{L_1} \varphi$
- $N, q \vDash_{L_1} \varphi_1 \vee \varphi_2$ *iff* $N, q \vDash_{L_1} \varphi_1$ *or* $N, q \vDash_{L_1} \varphi_2$.
- $N, q \vDash_{L_1} \Box_\alpha \varphi$ *iff* $N, q(\alpha) \vDash_{L_1} \varphi$
- $N, q \vDash_{L_1} [\mathsf{n}]\varphi$ *iff* $(N, q)[\mathsf{n}] \vDash_{L_1} \varphi$

From this definition is clear that a formula $[\mathsf{n}]\varphi$ should be read as: "after adding norm n to the normative system it is the case that φ holds". We have that the satisfaction relation $\vDash_{L_1}$ *extends* the satisfaction relation $\vDash_{L_0}$. That is, for every pointed normative systems (N, q) and formula $\varphi \in \mathcal{L}_0$ we have:

$$N, q \vDash_{L_1} \varphi \quad \Leftrightarrow \quad N, q \vDash_{L_0} \varphi$$

Thus, we can again simply write $\vDash$ instead of $\vDash_{L_1}$ or $\vDash_{L_0}$, since there should be no confusion between the intended satisfaction relation, i.e. for a formula $\varphi \in \mathcal{L}_0$ we can pick either, and for a formula $\varphi \in \mathcal{L}_1$ we pick $\vDash_{L_1}$.

Examples

Before we go to an axiomatization of this logic, let us look at some specific (non-)validities of this dynamic logic. First and foremost, if φ holds at a specific state, and we update with $(\varphi, +p, Act_R)$, we expect that the proposition p holds. This is reflected by the following validity:

$$\vDash \varphi \to [(\varphi, +p, Act_R)]p$$

We expect that the other way around is not necessarily the case (the right side implying the left side), since it might be that p was already the case in a state. Again, this is reflected by the following non-validity:

$$\nvDash ([(\varphi, +p, Act_R)]p) \to \varphi$$

However, if we know that $\neg p$ is the case and we know that after the update p holds, it can only be the case that φ holds. Thus:

$$\vDash \neg p \wedge ([(\varphi, +p, Act_R)]p) \to \varphi$$

For any sequence of actions $\alpha_1 \dots \alpha_n \in Act^*$ such that for a certain α_i $(1 \leq i \leq n)$ we have that $\alpha_i \in Act_R$, we have the following expected validity:

$$\vDash ([(\varphi, \pm p, Act_R)](\Box_{\alpha_1} \cdots \Box_{\alpha_n} \psi)) \leftrightarrow \Box_{\alpha_1} \cdots \Box_{\alpha_n} \psi$$

That is to say, whenever a norm effect is repaired, the truth of a formula ψ depends merely on whether it was true before the update; i.e. nothing changes. This follows from the fact that these updates are norm-aligned, as follows from Proposition 5.1. However, when the norm effect is not repaired, we can infer the truth of ψ by *first* performing the sequence of actions $\alpha_1 \dots \alpha_n$ and *then* updating the system. Or in short, it does not matter at what moment the update is performed. Thus, for any sequence of actions $\alpha_1 \dots \alpha_n \in Act^*$ such that $\alpha_1, \dots, \alpha_n \notin Act_R$ we have:

$$\vDash [(\varphi, \pm p, Act_R)](\Box_{\alpha_1} \dots \Box_{\alpha_n} \psi) \leftrightarrow (\Box_{\alpha_1} \dots \Box_{\alpha_n} [(\varphi, \pm p, Act_R)]\psi)$$

We also have the important property that order matters. First updating with norm n_0 and then with n_1 may have different results than first updating with n_1 and then with n_0. For example, we have:

$$\vDash [(\top, +p, \varnothing)]([(p, +p', \varnothing)]p') \text{ , and,}$$
$$\nvDash [(p, +p', \varnothing)]([(\top, +p, \varnothing)]p')$$

Lastly, we can also have updates without any effects, since for example the update $(\bot, +p, Act_R)$ leaves the truth of any formula unchanged:

$$\vDash ([(\bot, +p, Act_R)]\psi) \leftrightarrow \psi$$

We will now show how these properties lead to an axiomatization of our logic.

5.3.3 Axiomatization

In the previous section, we saw some example validities which we can use in this section to provide an axiomatization of our logic. But before we do this, we have to prove an important result first. We want to prove that for every pointed normative system (N, q), norm $\mathsf{n} \in \mathcal{N}_1$ and formula $\varphi \in \mathcal{L}_1$ that:

$$N[\mathsf{n}], q^r \vDash \varphi \quad \Leftrightarrow \quad N, q \vDash \varphi$$

In words, this equivalence states that the validity of φ in the updated normative system in *a repaired state* is equivalent to the validity of φ in the original system at the original state. To prove that this holds for *every formula* $\varphi \in \mathcal{L}_1$, we have to show that pointed system $(N[\mathsf{n}], q^r)$ and pointed system (N, q) behave in some sense 'similarly'. As shown in [Chellas, 1980], a way of doing this is by showing that the systems are *bi-similar*.

Definition 5.5 (Bi-simulation of normative systems)**.** *Given two normative systems* $N = (Q, \Pi, \mu, Act, \rightarrow)$ *and* $N' = (Q', \Pi', \mu', Act, \rightarrow')$*, we call a relation* $Z \subseteq Q \times Q'$ *a bi-simulation over* (N, N') *iff* $(q, q') \in Z$ *implies:*

1. $\mu(q) = \mu'(q')$*, and,*
2. *For every* $\alpha \in Act$*,* $(q(\alpha), q'(\alpha)) \in Z$*.*

Note that the second property is a simplification of the classical "zig" and "zag" properties, which we can make due to the underlying relations of the Kripke models being functional. A well-established result in the theory of modal logic is that if we have a bi-simulation Z over (N, N') for which $(q, q') \in Z$, then for any modal formula $\varphi \in \mathcal{L}_0$ we have:

$$N, q \vDash \varphi \quad \Leftrightarrow \quad N', q' \vDash \varphi$$

However, it is perhaps not immediately apparent that the same result holds whenever $\varphi \in \mathcal{L}_1$, i.e. whenever φ contains nested dynamic norm updates. Before we show that this is indeed the case, we need the following Lemma which shows that bi-simulation relations are (under suitable definition) preserved under dynamic update.

Lemma 5.1. *Let* N_0 *and* N_1 *be two normative systems, and let* Z *be a bi-simulation over* (N_0, N_1)*. Given norm* $\mathsf{n} = (\varphi, \pm p, Act_R) \in \mathcal{N}_1$*, let* $Z[\mathsf{n}]$ *be defined as follows:*

$$Z[\mathsf{n}] := \{(q_i^a, q_j^a), (q_i^r, q_j^r) \mid (q_i, q_j) \in Z\}$$

We have that $Z[\mathsf{n}]$ *is a bi-simulation over* $(N_0[\mathsf{n}], N_1[\mathsf{n}])$*.*

Proof. To see why this is the case, we verify both bi-simulation conditions separately. Let:

$$N_0 = (Q_0, \Pi, \mu_0, Act, \to_0) \quad \text{and} \quad N_0[\mathsf{n}] = N_0' = (Q_0', \Pi, \mu_0', Act, \to_0')$$
$$N_1 = (Q_1, \Pi, \mu_1, Act, \to_1) \quad \text{and} \quad N_1[\mathsf{n}] = N_1' = (Q_1', \Pi, \mu_1', Act, \to_1')$$

1. For the first bi-simulation condition, we assume an arbitrary $(q_i^x, q_j^x) \in Z[\mathsf{n}]$, where x can be either r or a:

   - If $x = r$, we want to show that $\mu_0'(q_i^r) = \mu_1'(q_j^r)$. We have by definition that $\mu_0'(q_i^r) = \mu_0(q_i)$. But since $(q_i, q_j) \in Z$, we have that $\mu_0(q_i) = \mu_1(q_j)$, and by definition $\mu_1(q_j) = \mu_1'(q_j^r)$, allowing us to conclude $\mu_0'(q_i^r) = \mu_1'(q_j^r)$.
   - If $x = a$, we want to show that $\mu_0'(q_i^a) = \mu_1'(q_j^a)$. Assume that $N_0, q_i \vDash \varphi$. In this case, we have that $\mu_0'(q_i^a) = \mu_0(q_i) \cup \{p\}$ (alternatively, $\mu_0(q_i) \backslash \{p\}$ in the case of $-p$). However, since $(q_i, q_j) \in Z$, we have that $\mu_0(q_i) = \mu_1(q_j)$, and since $\varphi \in \mathcal{L}_0$ also that $N_1, q_j \vDash \varphi$. This allows us to conclude that $\mu_1(q_j) \cup \{p\} = \mu_1'(q_j^a)$, and thus $\mu_0'(q_i^a) = \mu_1'(q_j^a)$ as needed. When $N_0, q_i \nvDash \varphi$, we have that $\mu_0'(q_i^a) = \mu_0(q_i)$, and due to $(q_i, q_j) \in Z$ we have again that $\mu_0(q_i) = \mu_1(q_j)$ and $N_1, q_j \nvDash \varphi$, and thus $\mu_1(q_j) = \mu_1'(q_j^a)$. This again allows us to conclude that $\mu_0'(q_i^a) = \mu_1'(q_j^a)$, as needed.

2. For the second bi-simulation condition, we again assume an arbitrary $(q_i^x, q_j^x) \in Z[\mathsf{n}]$ and an arbitrary $\alpha \in Act$, where x can be either r or a:

   - If $x = r$, we want to show that $(q_i^r(\alpha), q_j^r(\alpha)) \in Z[\mathsf{n}]$. Since we have that $(q_i(\alpha), q_j(\alpha)) \in Z$, we have by definition of $Z[\mathsf{n}]$ that $(q_i(\alpha)^r, q_j(\alpha)^r) \in Z[\mathsf{n}]$. But since $q_i(\alpha)^r = q_i^r(\alpha)$ and $q_j(\alpha)^r = q_j^r(\alpha)$, we have $(q_i^r(\alpha), q_j^r(\alpha)) \in Z[\mathsf{n}]$, as needed.
   - If $x = a$, we want to show that $(q_i^a(\alpha), q_j^a(\alpha)) \in Z[\mathsf{n}]$. Since we have that $(q_i(\alpha), q_j(\alpha)) \in Z$, we have by definition of $Z[\mathsf{n}]$ that $(q_i(\alpha)^r, q_j(\alpha)^r) \in Z[\mathsf{n}]$ and $(q_i(\alpha)^a, q_j(\alpha)^a) \in Z[\mathsf{n}]$. If $\alpha \in Act_R$, we have that $q_i(\alpha)^r = q_i^a(\alpha)$ and $q_j(\alpha)^r = q_j^a(\alpha)$, and thus $(q_i^a(\alpha), q_j^a(\alpha)) \in Z[\mathsf{n}]$, as needed. If $\alpha \notin Act_R$, we have that $q_i(\alpha)^a = q_i^a(\alpha)$ and $q_j(\alpha)^a = q_j^a(\alpha)$, and thus also $(q_i^a(\alpha), q_j^a(\alpha)) \in Z[\mathsf{n}]$, as needed.

□

Using this Lemma, we can establish the next important result, which shows that if two states are in a bi-simulation relation, they not only agree on all formulas from $\mathcal{L}_0$, but also on all formulas from $\mathcal{L}_1$.

Lemma 5.2. *Let N and N' be two normative systems, and let Z be a bi-simulation over (N, N'). Given that $(q_i, q_j) \in Z$, we have for every formula $\varphi \in \mathcal{L}_1$:*

$$N, q_i \vDash \varphi \quad \Leftrightarrow \quad N', q_j \vDash \varphi$$

Proof. If $(q_i, q_j) \in Z$, then they have to agree on all formulas from $\mathcal{L}_1$, i.e. all modal formulas including dynamic nested norm updates. We can prove this by induction on φ. Let our induction hypothesis be that if q_i and q_j are in a bi-simulation relation, they both agree on φ. The interesting case is when, from this assumption, we wish to conclude that this implies that they also both agree on $[\mathsf{n}]\varphi$. Assume $N, q_i \vDash [\mathsf{n}]\varphi$, from which we wish to show that this implies $N', q_j \vDash [\mathsf{n}]\varphi$. By our assumption that $N, q_i \vDash [\mathsf{n}]\varphi$, we have that $N[\mathsf{n}], q_i^a \vDash \varphi$. But, since $(q_i^a, q_j^a) \in Z[\mathsf{n}]$ where $Z[\mathsf{n}]$ is a bi-simulation over $(N[\mathsf{n}], N'[\mathsf{n}])$ as defined as in Lemma 5.1, we can use our induction hypothesis to conclude that $N'[\mathsf{n}], q_j^a \vDash \varphi$, and thus that $N, q_j \vDash [\mathsf{n}]\varphi$ as needed. The other direction is similar. □

An important consequence of this Lemma is that any formula from the set $\mathcal{L}_1$ is (equivalent to) a modal formula, which is due to van Benthem's characterization theorem (also known as the Modal Invariance Theorem) which states that any formula which cannot make a distinction between bi-similar models has this property, see [Chellas, 1980]. We now have developed sufficient tools to solve our earlier mentioned problem.

Lemma 5.3. *For every pointed normative system (N, q), norm $\mathsf{n} \in \mathcal{N}_1$ and formula $\varphi \in \mathcal{L}_1$, we have:*

$$N[\mathsf{n}], q^r \vDash \varphi \quad \Leftrightarrow \quad N, q \vDash \varphi$$

Proof. Let $N = (Q, \Pi, \mu, Act, \rightarrow)$ and $N[\mathsf{n}] = (Q', \Pi, \mu', Act, \rightarrow')$. From Lemma 5.2 we know that it is sufficient to define a bi-simulation relation Z over $(N, N[\mathsf{n}])$ for which $(q, q^r) \in Z$. Define $Z = \{(q, q^r) \mid q \in Q\}$. The first bi-simulation condition holds, since for all $q \in Q$ we have that $\mu'(q^r) = \mu(q)$ by definition. To show the second bi-simulation condition, assume an arbitrary $q \in Q$ and $\alpha \in Act$. Since $(q, q^r) \in Z$, we wish to show that $(q(\alpha), q^r(\alpha)) \in Z$. Since $(q(\alpha), q(\alpha)^r) \in Z$ by definition, and since $q(\alpha)^r = q^r(\alpha)$, this trivially holds. The last observation we need to make is that for initial state q we have that $(q, q^r) \in Z$, as needed. □

In words, this lemma states that evaluating any formula in an updated system at a repaired state is equivalent to evaluating this formula in the original system at the original state. Or, when the norm effect has been repaired, the behaviour of the system is equivalent to what it was before the update.

An important rule that we need for our axiomatization is the rule of replacement of equivalent formulas (RE), given by:

$$\frac{\varphi_1 \leftrightarrow \varphi_2}{\psi \leftrightarrow \psi[\varphi_1/\varphi_2]} \text{ (RE)}$$

The notation $\psi[\varphi_1/\varphi_2]$ denotes formula ψ where zero or more occurrences of φ_2 are replaced by φ_1. This rule states that if we can prove that φ_1 and φ_2 are logically equivalent, then we can replace occurrences of φ_2 with φ_1 within a formula ψ to obtain a logically equivalent formula to ψ. Although this rule is sound for logical formulas from $\mathcal{L}_0$, it is not immediately apparent that this is also the case for formulas from $\mathcal{L}_1$, e.g. a natural question to ask is whether this rule is still sound whenever a replacement occurs within the scope of a dynamic norm update. The following lemma shows that it is indeed the case that this rule is sound in $\mathcal{L}_1$.

Lemma 5.4. *Rule for replacement of equivalent formulas (RE) is sound for $\mathcal{L}_1$. That is, $\forall \varphi_1, \varphi_2, \psi \in \mathcal{L}_1$ we have:*

$$\vDash_{L_1} (\varphi_1 \leftrightarrow \varphi_2) \quad \Rightarrow \quad \vDash_{L_1} \psi \leftrightarrow \psi[\varphi_1/\varphi_2]$$

Proof. We prove it by induction on ψ. Let $\vDash_{L_1} \psi \leftrightarrow \psi[\varphi_1/\varphi_2]$ be our induction hypothesis. The interesting case occurs when we want to show that this implies $\vDash_{L_1} [\mathsf{n}]\psi \leftrightarrow [\mathsf{n}](\psi[\varphi_1/\varphi_2])$, i.e. whenever a replacement occurs in the scope of a dynamic norm update. Assume an arbitrary pointed normative system (N, q) for which $N, q \vDash_{L_1} [\mathsf{n}]\psi$. From this, we have that $(N, q)[\mathsf{n}] \vDash_{L_1} \psi$, and from our induction hypothesis we have that $(N, q)[\mathsf{n}] \vDash_{L_1} \psi[\varphi_1/\varphi_2]$. This allows us to conclude $N, q \vDash_{L_1} [\mathsf{n}](\psi[\varphi_1/\varphi_2])$, and since (N, q) was chosen arbitrarily also $\vDash_{L_1} [\mathsf{n}](\psi[\varphi_1/\varphi_2])$, as needed. The other direction is similar. □

The underlying idea that we use in the above proof is that the result of a dynamic norm update on a normative system *always* results in a normative system, i.e. any result we acquire for *all* normative systems also applies to arbitrary updated normative systems.

We are now ready to give the axiomatization of $\mathcal{L}_1$, which is given by System **L1** shown in Table 5.1. As standard, we write $\vdash_{L_1} \varphi$ whenever a formula $\varphi \in \mathcal{L}_1$ is provable from the rules and axioms from **L1**. Axiom 1.1(a) states that if we update every φ-state with $+p$, in order for p to be true it either had to be true before the update, or φ is the case. Axiom 1.1(b) states that if we update every φ-state with $-p$, in order for p to be true it has to be both true before the update and the state itself was not updated, i.e. $\neg\varphi$ is the case. Axiom 1.2 states that any other proposition for which the update does not apply behave invariantly: i.e. the truth condition remains as what it was

Axiom schemata	
1.1(a).	$([(\varphi, +p, Act_R)]p) \leftrightarrow (\varphi \vee p)$
1.1(b).	$([(\varphi, -p, Act_R)]p) \leftrightarrow (\neg\varphi \wedge p)$
1.2.	$([\mathsf{n}]p') \leftrightarrow p'$ (if $p \neq p'$)
1.3.	$([\mathsf{n}]\neg\psi) \leftrightarrow (\neg[\mathsf{n}]\psi)$
1.4.	$([\mathsf{n}](\psi_1 \vee \psi_2)) \leftrightarrow (([\mathsf{n}]\psi_1) \vee ([\mathsf{n}]\psi_2))$
1.5.	$([\mathsf{n}]\,\Box_\alpha\,\psi) \leftrightarrow (\Box_\alpha\psi)$ (if $\alpha \in Act_R$)
1.6.	$([\mathsf{n}]\,\Box_\alpha\,\psi) \leftrightarrow (\Box_\alpha[\mathsf{n}]\psi)$ (if $\alpha \notin Act_R$)
	+ All axioms and rules from system **L0**, including the rule for replacement of equivalent formulas (RE).

Table 5.1: System **L1***, where* $\mathsf{n} = (\varphi, \pm p, Act_R) \in \mathcal{N}_1$.

before the update. Axiom 1.3 and 1.4 both state that the dynamic update is a *function*: a normative system N together with an update n uniquely defines an updated system $N[\mathsf{n}]$. Axiom 1.5 states the repair property we saw earlier, i.e. whenever a repair action is performed, the norm is no longer in effect. Finally, axiom 1.6 states the property that if a certain action does not repair the norm effect, it does not matter if we perform the update before or after this action in order to determine the state of affairs caused by this action. We have the following important result.

Theorem 5.2 (Soundness of **L1**)**.** *System* **L1***, as shown in Table 5.1, is sound with respect to the corresponding semantics of logic* $\mathcal{L}_1$*. That is, for every formula* $\varphi \in \mathcal{L}_1$ *we have:*

$$\vdash_{L_1} \varphi \quad \Rightarrow \quad \vDash_{L_1} \varphi$$

Proof. Soundness of rule (RE) is shown in Lemma 5.4. Soundness of the remaining axioms can be proven by proving soundness for every axiom independently. Every step in the proof is some application of a definition found earlier in this chapter, except for axiom 1.5 where we use the important result acquired in Lemma 5.3 (denoted with $\overset{!}{\Leftrightarrow}$):

1.1(a).

$$\begin{aligned} N, q \vDash [(\varphi, +p, Act_R)]p \quad &\Leftrightarrow \quad N[(\varphi, +p, Act_R)], q^a \vDash p \quad \Leftrightarrow \\ p \in \mu'(q^a) \quad &\Leftrightarrow \quad N, q \vDash \varphi \text{ or } p \in \mu(q) \quad \Leftrightarrow \\ & N, q \vDash \varphi \vee p \end{aligned}$$

1.1(b).

$$\begin{aligned} N, q \vDash [(\varphi, -p, Act_R)]p \quad &\Leftrightarrow \quad N[(\varphi, -p, Act_R)], q^a \vDash p \quad \Leftrightarrow \\ p \in \mu'(q^a) \quad &\Leftrightarrow \quad N, q \vDash \neg\varphi \text{ and } p \in \mu(q) \quad \Leftrightarrow \\ & N, q \vDash \neg\varphi \wedge p \end{aligned}$$

1.2.

$$\begin{array}{ccc} N,q \vDash [\mathsf{n}]p' & \Leftrightarrow & N[\mathsf{n}],q^a \vDash p' & \Leftrightarrow \\ p' \in \mu'(q^a) & \Leftrightarrow & p' \in \mu(q) \text{ (if } p \neq p') & \Leftrightarrow \\ \multicolumn{3}{c}{N,q \vDash p' \text{ (if } p \neq p')} \end{array}$$

1.3.

$$\begin{array}{ccc} N,q \vDash [\mathsf{n}]\neg\psi & \Leftrightarrow & (N,q)[\mathsf{n}] \vDash \neg\psi & \Leftrightarrow \\ (N,q)[\mathsf{n}] \nvDash \psi & \Leftrightarrow & N,q \nvDash [\mathsf{n}]\psi & \Leftrightarrow \\ \multicolumn{3}{c}{N,q \vDash \neg[\mathsf{n}]\psi} \end{array}$$

1.4.

$$\begin{array}{ccc} N,q \vDash [\mathsf{n}](\psi_1 \vee \psi_2) & \Leftrightarrow & (N,q)[\mathsf{n}] \vDash \psi_1 \vee \psi_2 & \Leftrightarrow \\ (N,q)[\mathsf{n}] \vDash \psi_1 \text{ or } (N,q)[\mathsf{n}] \vDash \psi_2 & & & \Leftrightarrow \\ N,q \vDash [\mathsf{n}]\psi_1 \text{ or } N,q \vDash [\mathsf{n}]\psi_2 & & & \Leftrightarrow \\ \multicolumn{3}{c}{N,q \vDash ([\mathsf{n}]\psi_1) \vee ([\mathsf{n}]\psi_2)} \end{array}$$

1.5.

$$\begin{array}{ccc} N,q \vDash [\mathsf{n}]\,\Box_\alpha\,\psi & \Leftrightarrow & N[\mathsf{n}],q^a \vDash \Box_\alpha\psi & \Leftrightarrow \\ \multicolumn{3}{c}{N[\mathsf{n}],q^a(\alpha) \vDash \psi} & \Leftrightarrow \\ \multicolumn{3}{c}{N[\mathsf{n}],q(\alpha)^r \vDash \psi \text{ (if } \alpha \in Act_R)} & \overset{!}{\Leftrightarrow} \\ \multicolumn{3}{c}{N,q(\alpha) \vDash \psi \text{ (if } \alpha \in Act_R)} & \Leftrightarrow \\ \multicolumn{3}{c}{N,q \vDash \Box_\alpha\psi \text{ (if } \alpha \in Act_R)} \end{array}$$

1.6.

$$\begin{array}{ccc} N,q \vDash [\mathsf{n}]\,\Box_\alpha\,\psi & \Leftrightarrow & N[\mathsf{n}],q^a \vDash \Box_\alpha\psi & \Leftrightarrow \\ \multicolumn{3}{c}{N[\mathsf{n}],q^a(\alpha) \vDash \psi} & \Leftrightarrow \\ \multicolumn{3}{c}{N[\mathsf{n}],q(\alpha)^a \vDash \psi \text{ (if } \alpha \notin Act_R)} & \Leftrightarrow \\ \multicolumn{3}{c}{N,q(\alpha) \vDash [\mathsf{n}]\psi \text{ (if } \alpha \notin Act_R)} & \Leftrightarrow \\ \multicolumn{3}{c}{N,q \vDash \Box_\alpha[\mathsf{n}]\psi \text{ (if } \alpha \notin Act_R)} \end{array}$$

□

Just like in dynamic epistemic logic, these rules and axioms allow for reduction. Particularly, these rules and axioms allow us to reduce formulas of $\mathcal{L}_1$ to formulas of $\mathcal{L}_0$, which allows us to show completeness. This important result is shown in the next Theorem.

Theorem 5.3 (Completeness of **L1**)**.** *System* **L1**, *as shown in Table 5.1, is complete with respect to the corresponding semantics of logic* $\mathcal{L}_1$*. That is, for every formula* $\varphi \in \mathcal{L}_1$ *we have:*

$$\vDash_{L_1} \varphi \quad \Rightarrow \quad \vdash_{L_1} \varphi$$

Proof. In order to show completeness, we define the following translation function $\tau_1 : \mathcal{L}_1 \to \mathcal{L}_0$:

$$\begin{array}{rcll}
\tau_1(p) & = & p & \\
\tau_1([(\varphi, +p, Act_R)]p) & = & \varphi \vee p & \\
\tau_1([(\varphi, -p, Act_R)]p) & = & \neg\varphi \wedge p & \\
\tau_1(\neg\psi) & = & \neg\tau_1(\psi) & \\
\tau_1(\psi_1 \vee \psi_2) & = & \tau_1(\psi_1) \vee \tau_1(\psi_2) & \\
\tau_1(\Box_\alpha\psi) & = & \Box_\alpha\tau_1(\psi) & \\
\tau_1([\mathsf{n}]\neg\psi) & = & \tau_1(\neg[\mathsf{n}]\psi) & \\
\tau_1([\mathsf{n}](\psi_1 \vee \psi_2)) & = & \tau_1(([\mathsf{n}]\psi_1) \vee ([\mathsf{n}]\psi_2)) & \\
\tau_1([\mathsf{n}]\,\Box_\alpha\,\psi) & = & \tau_1(\Box_\alpha\psi) & (\text{if } \alpha \in Act_R) \\
\tau_1([\mathsf{n}]\,\Box_\alpha\,\psi) & = & \tau_1(\Box_\alpha[\mathsf{n}]\psi) & (\text{if } \alpha \notin Act_R) \\
\tau_1([\mathsf{n}_1]([\mathsf{n}_2]\psi)) & = & \tau_1([\mathsf{n}_1]\tau_1([\mathsf{n}_2]\psi)) &
\end{array}$$

The first observation we can make is that this function is well-defined, i.e. for any formula $\varphi \in \mathcal{L}_1$ this function returns a a formula $\tau_1(\varphi) \in \mathcal{L}_0$. The reason that this function is well-defined (i.e. always gives an answer) is that we can show under a suitable definition of the complexity of formulas (see [van Ditmarsch et al., 2007]) that this translation always reduces the complexity. In order to show completeness, we proceed in the following ways. In (1), we show that this translation has the property that for any $\varphi \in \mathcal{L}_1$ we have that $\vdash_{\mathrm{L}_1} [\mathsf{n}]\tau_1(\varphi) \leftrightarrow \tau_1([\mathsf{n}]\varphi)$. In (2), we show that from (1) we can conclude that for any $\varphi \in \mathcal{L}_1$ we have $\vdash_{\mathrm{L}_1} \varphi \leftrightarrow \tau_1(\varphi)$. In (3) we prove our completeness claim using the result from (2).

1. The first step in the completeness proof is to show that for any $\varphi \in \mathcal{L}_1$ that $\vdash_{\mathrm{L}_1} [\mathsf{n}]\tau_1(\varphi) \leftrightarrow \tau_1([\mathsf{n}]\varphi)$. We show this by induction on the length of φ. Let $\vdash_{\mathrm{L}_1} [\mathsf{n}]\tau_1(\varphi_1) \leftrightarrow \tau_1([\mathsf{n}]\varphi_1)$ and $\vdash_{\mathrm{L}_1} [\mathsf{n}]\tau_1(\varphi_2) \leftrightarrow \tau_1([\mathsf{n}]\varphi_2)$ be our induction hypotheses. We prove from this assumption that (a) $\vdash_{\mathrm{L}_1} [\mathsf{n}]\tau_1(\varphi_1 \vee \varphi_2) \leftrightarrow \tau_1([\mathsf{n}](\varphi_1 \vee \varphi_2))$, (b) $\vdash_{\mathrm{L}_1} [\mathsf{n}]\tau_1(\Box_\alpha\varphi_1) \leftrightarrow \tau_1([\mathsf{n}](\Box_\alpha\varphi_1))$, and leave the other logical connectives (and the base cases) to the reader.

   (a) Due to axiom 1.4 we have $\vdash_{\mathrm{L}_1} [\mathsf{n}](\tau_1(\varphi_1) \vee \tau_1(\varphi_2)) \leftrightarrow [\mathsf{n}](\tau_1(\varphi_1)) \vee [\mathsf{n}](\tau_1(\varphi_2))$. By the rule of replacement of equivalent formulas (RE) from our induction hypotheses we get the result that $\vdash_{\mathrm{L}_1} [\mathsf{n}](\tau_1(\varphi_1) \vee \tau_1(\varphi_2)) \leftrightarrow \tau_1([\mathsf{n}]\varphi_1) \vee \tau_1([\mathsf{n}]\varphi_2)$. But since $\tau_1([\mathsf{n}]\varphi_1) \vee \tau_1([\mathsf{n}]\varphi_2) = \tau_1([\mathsf{n}]\varphi_1 \vee [\mathsf{n}]\varphi_2) = \tau_1([\mathsf{n}](\varphi_1 \vee \varphi_2))$ (from the definition of τ_1), we have $\vdash_{\mathrm{L}_1} [\mathsf{n}](\tau_1(\varphi_1) \vee \tau_1(\varphi_2)) \leftrightarrow \tau_1([\mathsf{n}](\varphi_1 \vee \varphi_2))$, as needed.

   (b) We either have that $\alpha \in Act_R$ or not. In the first case, we have due to axiom 1.5 and the definition of τ_1 that $\vdash_{\mathrm{L}_1} [\mathsf{n}](\tau_1(\Box_\alpha\varphi_1)) \leftrightarrow \tau_1(\Box_\alpha\varphi_1)$, but since $\tau_1(\Box_\alpha\varphi_1) = \tau_1([\mathsf{n}]\,\Box_\alpha\,\varphi_1)$ whenever $\alpha \in Act_R$

(from the definition of τ_1), we have $\vdash_{L_1} [\mathsf{n}](\tau_1(\Box_\alpha\varphi_1)) \leftrightarrow \tau_1([\mathsf{n}]\,\Box_\alpha\varphi_1))$ as needed. Whenever $\alpha \notin Act_R$, we have due to axiom 1.6 and the definition of τ_1 that $\vdash_{L_1} ([\mathsf{n}]\tau_1(\Box_\alpha\varphi_1)) \leftrightarrow (\Box_\alpha[\mathsf{n}]\tau_1(\varphi_1))$. By the rule of replacement of equivalent formulas (RE) from our induction hypotheses we get $\vdash_{L_1} ([\mathsf{n}]\tau_1(\Box_\alpha\varphi_1)) \leftrightarrow (\Box_\alpha\tau_1([\mathsf{n}]\varphi_1))$. But since $\Box_\alpha\tau_1([\mathsf{n}]\varphi_1) = \tau_1(\Box_\alpha[\mathsf{n}]\varphi_1) = \tau_1([\mathsf{n}]\,\Box_\alpha\,\varphi_1)$ whenever $\alpha \notin Act_R$ (from the definition of τ_1), we have $\vdash_{L_1} [\mathsf{n}](\tau_1(\Box_\alpha\varphi_1)) \leftrightarrow \tau_1([\mathsf{n}](\Box_\alpha\varphi_1))$ as needed.

2. The second step in the proof is to conclude that for any $\varphi \in \mathcal{L}_1$ we have that $\vdash_{L_1} \varphi \leftrightarrow \tau_1(\varphi)$. We can again show this by induction on the length of φ. Let $\vdash_{L_1} \varphi \leftrightarrow \tau_1(\varphi)$ be our induction hypothesis. The interesting case is when we want to show that this implies $\vdash_{L_1} [\mathsf{n}]\varphi \leftrightarrow \tau_1([\mathsf{n}]\varphi)$. We already established that $\vdash_{L_1} [\mathsf{n}]\tau_1(\varphi) \leftrightarrow \tau_1([\mathsf{n}]\varphi)$, allowing us to apply rule the rule of replacement of equivalent formulas (RE) from our induction hypothesis to achieve $\vdash_{L_1} [\mathsf{n}]\varphi \leftrightarrow \tau_1([\mathsf{n}]\varphi)$. We leave the other logical connectives to the reader.

3. The third and final step is to show our completeness claim, i.e. from the assumption that $\vDash_{L_1} \varphi$ we have $\vdash_{L_1} \varphi$. By soundness of $\mathcal{L}_1$ (Theorem 5.2), we have from $\vdash_{L_1} \varphi \leftrightarrow \tau_1(\varphi)$ that $\vDash_{L_1} \varphi \leftrightarrow \tau_1(\varphi)$. Thus, from our assumption that $\vDash_{L_1} \varphi$, we have $\vDash_{L_1} \tau_1(\varphi)$. By the completeness of system **L0** (Theorem 5.1), we have $\vdash_{L_0} \tau_1(\varphi)$. This implies that also $\vdash_{L_1} \tau_1(\varphi)$, since **L1** contains all the rules and axioms of **L0**. Since we have shown that $\vdash_{L_1} \varphi \leftrightarrow \tau_1(\varphi)$ and since $\vdash_{L_1} \tau_1(\varphi)$, we can apply rule MP to acquire $\vdash_{L_1} \varphi$.

□

This concludes our result on soundness and completeness of system **L1** for logic $\mathcal{L}_1$. In the next section we show how we can apply this system to deduce the validity of assertions from this language. Moreover, we show that this logic is decidable, i.e. there exists an effective procedure that decides whether a formula is valid or not.

5.3.4 Derivations and decidability

In this section we show how we can use system **L1** in order to derive the validity of assertions from language $\mathcal{L}_1$. Moreover, we show that this procedure is decidable. In Section 5.3.2 we already briefly mentioned without proof that the following assertion is valid:

$$\vDash_{L_1} [(\top, +p, \varnothing)]([(p, +p', \varnothing)]p')$$

Below, we prove it formally by providing a derivation from system **L1**. We use notation 'RE(x,y)' to denote that we replaced a sub-formula from line x by using an acquired equivalence from line y, which is short-hand notation for applying the rule of replacement of equivalence formulas (RE) and then applying modus ponens (MP) on the result. Moreover, we use the notation 'MP(x,y)' to denote that we apply modus ponens (MP) with the implication from line x and the precedent from line y:

1.	$([(p, +p', \varnothing)]p') \leftrightarrow (p \vee p')$	Ax.1.1(a)
2.	$([(\top, +p, \varnothing)](p \vee p')) \leftrightarrow ([(\top, +p, \varnothing)]p \vee [(\top, +p, \varnothing)]p')$	Ax.1.4
3.	$([(\top, +p, \varnothing)]p) \leftrightarrow (\top \vee p)$	Ax.1.1(a)
4.	$([(\top, +p, \varnothing)]p') \leftrightarrow p'$	Ax.1.2
5.	$([(\top, +p, \varnothing)](p \vee p')) \leftrightarrow ((\top \vee p) \vee [(\top, +p, \varnothing)]p')$	RE(2,3)
6.	$([(\top, +p, \varnothing)](p \vee p')) \leftrightarrow ((\top \vee p) \vee p')$	RE(5,4)
7.	$([(\top, +p, \varnothing)]([(p, +p', \varnothing)]p')) \leftrightarrow ((\top \vee p) \vee p')$	RE(6,1)
8.	$((\top \vee p) \vee p')$	Taut.
9.	$[(\top, +p, \varnothing)]([(p, +p', \varnothing)]p')$	MP(7,8)

Thus, we have:

$$\vdash_{\mathrm{L}_1} [(\top, +p, \varnothing)]([(p, +p', \varnothing)]p')$$

And by soundness of **L1** we have:

$$\vDash_{\mathrm{L}_1} [(\top, +p, \varnothing)]([(p, +p', \varnothing)]p')$$

A natural question we may ask is whether logic $\mathcal{L}_1$ with corresponding system **L1** is *decidable*, i.e. whether there exists an effective procedure that tells us whether an arbitrary formula $\varphi \in \mathcal{L}_1$ is valid. We have the following result.

Theorem 5.4. *The logical language $\mathcal{L}_1$ with corresponding system* **L1** *is decidable.*

Proof. We use the translation function $\tau_1 : \mathcal{L}_1 \to \mathcal{L}_0$ from our completeness proof in Theorem 5.2 to establish the following correspondence:

$$\vdash_{\mathrm{L}_1} \varphi \quad \Leftrightarrow \quad \vdash_{\mathrm{L}_1} \tau_1(\varphi) \quad \Leftrightarrow \quad \vdash_{\mathrm{L}_0} \tau_1(\varphi)$$

Thus, in order to determine whether $\vdash_{\mathrm{L}_1} \varphi$, it is both necessary and sufficient to show that $\vdash_{\mathrm{L}_0} \tau_1(\varphi)$. However, since logic $\mathcal{L}_0$ with corresponding system **L0** is decidable and since τ_1 is well defined and can be effectively computed, decidability immediately transfers to our logic and system. □

This proof not only gives the result that our logic is decidable, it also highlights an easier way to establish validity of a sentence. For example, to

show the following non-validity taken from Section 5.3.2 (notice that the order in which the updates occur is reversed):

$$\nvdash_{\mathrm{L}_1} [(p, +p', \varnothing)]([(\top, +p, \varnothing)]p')$$

We can apply our translation function τ_1 on $[(p, +p', \varnothing)]([(\top, +p, \varnothing)]p')$:

$$\begin{array}{rcl} \tau_1([(p, +p', \varnothing)]([(\top, +p, \varnothing)]p')) & = & \tau_1([(p, +p', \varnothing)]\tau_1([(\top, +p, \varnothing)]p')) = \\ \tau_1([(p, +p', \varnothing)]p') & = & p \vee p' \end{array}$$

To easily establish that:

$$\nvdash_{\mathrm{L}_0} p \vee p'$$

Similarly, we can apply this translation to $[(\top, +p, \varnothing)]([(p, +p', \varnothing)]p')$:

$$\begin{array}{rcl} \tau_1([(\top, +p, \varnothing)]([(p, +p', \varnothing)]p')) & = & \tau_1([(\top, +p, \varnothing)]\tau_1([(p, +p', \varnothing)]p')) = \\ \tau_1([(\top, +p, \varnothing)]\tau_1(p \vee p')) & = & \tau_1([(\top, +p, \varnothing)](\tau_1(p) \vee \tau_1(p'))) = \\ \tau_1([(\top, +p, \varnothing)](p \vee p')) & = & \tau_1([(\top, +p, \varnothing)]p \vee [(\top, +p, \varnothing)]p') = \\ \tau_1([(\top, +p, \varnothing)]p) \vee \tau_1([(\top, +p, \varnothing)]p') & = & (\top \vee p) \vee p' \end{array}$$

To easily establish that:

$$\vdash_{\mathrm{L}_0} (\top \vee p) \vee p'$$

This concludes our work on the logical language $\mathcal{L}_1$ and **L1** allowing us to reason about normative update of the 'to-be' variant. In the next section we take a look at normative update of the 'to-do' variant, leading to another dynamic logic which, as we will see later, will extend this logic.

5.4 Extended language for norm update

In the previous section, we developed a basic logical language of normative update. In this section we consider a different kind of normative update, which corresponds to norms of the 'to-do' variant, which we mentioned in Section 5.2.4. A norm of this class, when added to the system, causes that after the performance of certain actions in certain states certain soft facts become true or false, until a repair action occurs.

In the remainder of this section, we give a language to construct these kinds of norms, we show how we can update a normative system using these norms and finally we show how we can add these constructs to our logical language.

5.4.1 Language and update

Given a normative system $N = (Q, \Pi, \mu, Act, \rightarrow)$, we construct the norm language $\mathcal{N}_2$ in the following way, where $\varphi \in \mathcal{L}_0$, $p \in \Pi_{\text{soft}}$, $Act_T \subseteq Act$ and $Act_R \subseteq Act$:

$$\mathsf{n} ::= (Act_T, \varphi, +p, Act_R) \mid (Act_T, \varphi, -p, Act_R)$$

From the definition of this grammar, it should be apparent that $\mathcal{N}_1 \cap \mathcal{N}_2 = \varnothing$. However, note that we use the same symbol n for constructs of the language $\mathcal{N}_1$ and $\mathcal{N}_2$, so we should always make the language we are referring to explicit. For a given update $(Act_T, \varphi, \pm p, Act_R)$ (where $\pm p$ can again be either $+p$ or $-p$), we call the set Act_T the set of trigger actions, and again refer to φ as the norm condition, $\pm p$ as the norm effect and Act_R as the repair actions. Adding a norm $(Act_T, \varphi, +p, Act_R)$ implies that whenever a trigger action from Act_T occurs, for every φ-state the soft proposition p will start to hold, until a repair action from the set Act_R occurs. An important distinction with norms from $\mathcal{N}_1$ is that norms from $\mathcal{N}_2$ stay 'active', i.e. whenever a repair action occurs which repairs the norm effect, the norm effect may become triggered again by a trigger action. For example, the norm $(\{\alpha\}, \top, +v, Act \backslash \{\alpha\})$ states that always after the performance of action α the violation v will start to hold, which is repaired immediately afterwards by all actions (except by α itself). Again, since we want to give a clear semantic interpretation to how a pointed normative system should behave when a norm is added to the system, we extend the notion of *norm-aligned* to updates from this extended language.

Postulate 5.2 (Norm-Aligned updates from $\mathcal{N}_2$)**.** *Any normative system updated with a norm* $\mathsf{n} \in \mathcal{N}_2$ *should be norm-aligned with* n*. Given a pointed normative system* (N, q) *and* $\mathsf{n} \in \mathcal{N}_2$*, let* $(N, q)'$ *be the system updated with* n*. We distinguish between the cases where* $\mathsf{n} = (Act_T, \varphi, +p, Act_R)$ *or where* $\mathsf{n} = (Act_T, \varphi, -p, Act_R)$*. We say that* $(N, q)'$ *is* norm-aligned *with ...*

1. *...* $(Act_T, \varphi, +p, Act_R)$ *iff for every atomic proposition* $p' \in \Pi$ *and every (possibly empty) finite sequence of actions* $\alpha_1 \dots \alpha_n \in Act^*$*, we have that* $(N, q)' \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p'$ *if and only if:*

   (a) $N, q \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p'$, ***or;***

   (b) $p = p'$, $N, q \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} \varphi$, *and,*

   $$\exists i \in \{1, \dots, n\} : (\alpha_i \in Act_T \text{ and } \forall j \in \{i+1, \dots, n\} : \alpha_j \notin Act_R)$$

2. *...* $(Act_T, \varphi, -p, Act_R)$ *iff for every atomic proposition* $p' \in \Pi$ *and every (possibly empty) finite sequence of actions* $\alpha_1 \dots \alpha_n \in Act^*$*, we have that* $(N, q)' \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p'$ *if and only if:*

   (a) $N, q \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p'$, ***and;***

(b) $p \neq p'$, *or,* $N, q \not\models \Box_{\alpha_1} \cdots \Box_{\alpha_n} \varphi$, *or,*

$$\forall i \in \{1, \ldots, n\} : (\alpha_i \in Act_T \Rightarrow \exists j \in \{i+1, \ldots, n\} : \alpha_j \in Act_R)$$

In words, after an update $(Act_T, \varphi, +p, Act_R)$, action $\alpha_1 \ldots \alpha_n$ should bring about p' in the updated system iff either (1) $\alpha_1 \ldots \alpha_n$ already brought about p' in the original system before the update $(N, q \models \Box_{\alpha_1} \cdots \Box_{\alpha_n} p')$, or (2) p' is added by the norm effect $(p' = p)$, the norm condition is brought about by $\alpha_1 \ldots \alpha_n$ $(N, q \models \Box_{\alpha_1} \cdots \Box_{\alpha_n} \varphi)$, and the norm effect was *triggered* and *not already repaired* $(\exists i \in \{1, \ldots, n\} : (\alpha_i \in Act_T$ and $\forall j \in \{i+1, \ldots, n\} : \alpha_j \notin Act_R))$. Notice that this second requirement is completely different from the second requirement we saw in Postulate 5.1. Again, this notion captures both a property of success and a property of minimal change. On the one hand, it states that the norm effect should hold under the right conditions (property of success corresponding to item 2), and on the other hand it states that everything else should remain unchanged (property of minimal change corresponding to item 1). Notice that by this interpretation of norm-aligned, whenever we have a norm $(Act_T, \varphi, \pm p, Act_R)$ and an action α such that $\alpha \in Act_T$ *and* $\alpha \in Act_R$, it is always the case that action α triggers the norm effect and *not* repairs it, even though this action is both a trigger and a repair action. In other words, the trigger actions take priority over the repair actions. Thus, by this interpretation, adding a norm $(Act_T, \varphi, \pm p, Act_R)$ or a norm $(Act_T, \varphi, \pm p, Act_R \backslash Act_T)$ would cause the updated system to have the same behaviour. Again, more complex norms can be encoded by combining a multitude of these simpler norms.

We will now show how we can update a pointed normative system to a new pointed normative system with our new norms. This update is very similar to the previous update we saw in this chapter, the most important difference lying in the updated accessibility relation of the model. We will show later that this update will result in a norm-aligned system. Given a pointed normative system (N, q) and a norm $\mathsf{n} \in \mathcal{N}_2$, we again write $(N, q)[\mathsf{n}]$ to denote the updated system.

Definition 5.6 (Updates from $\mathcal{N}_2$)**.** *Given* $N = (Q, \Pi, \mu, Act, \rightarrow)$, $q \in Q$ *and* $\mathsf{n} = (Act_T, \varphi, +p, Act_R) \in \mathcal{N}_2$, *we let* $(N, q)[\mathsf{n}] = (N[\mathsf{n}], q[\mathsf{n}])$ *such that:*

- $N[\mathsf{n}] = (Q', \Pi, \mu', Act, \rightarrow')$, *where:*
  - $Q' = \{q^r, q^a \mid q \in Q\}$
  - *For every* $q \in Q$*:*

$$\mu'(q^r) = \mu(q) \quad \textit{and} \quad \mu'(q^a) = \begin{cases} \mu(q) \cup \{p\} & \textit{if } N, q \models \varphi \\ \mu(q) & \textit{otherwise} \end{cases}$$

$$
\begin{aligned}
- \quad \rightarrow' = \quad & \{(q_i^r, \alpha, q_j^r) \mid q_i(\alpha) = q_j \ \textit{and}\ \alpha \notin Act_T\} \cup \\
& \{(q_i^r, \alpha, q_j^a) \mid q_i(\alpha) = q_j \ \textit{and}\ \alpha \in Act_T\} \cup \\
& \{(q_i^a, \alpha, q_j^a) \mid q_i(\alpha) = q_j \ \textit{and}\ \alpha \notin (Act_R \backslash Act_T)\} \cup \\
& \{(q_i^a, \alpha, q_j^r) \mid q_i(\alpha) = q_j \ \textit{and}\ \alpha \in (Act_R \backslash Act_T)\}
\end{aligned}
$$

- *State* $q[\mathsf{n}] = q^r$

The update with $-p$ works analogously, except we remove this atomic proposition from the updated valuation function μ' for every active state. Again, for every state $q \in Q$ we create two copies in the updated system; one in which the norm effect is active and one in which it is repaired. There are two important differences in updates from $\mathcal{N}_2$ in comparison with updates from $\mathcal{N}_1$. They are the following:

1. The updated accessibility relation is different. It is still the case that whenever we are in an active state and a repair action has been performed (which is not also a trigger action), we go to a repaired state. However, whenever we are in a repaired state and a trigger action has been performed, we go to an active state. In other words, with this update we are both able to go from active states to repaired states and vice-versa. If no trigger or repair action is performed, we remain in an active (or repaired) state.

2. The updated current state is different. With this update the current state is updated to a repaired state, while in updates from $\mathcal{N}_1$ we updated the current state to an active state.

The following proposition shows that these updates again result in a norm-aligned system. Note that we only show $(Act_T, \varphi, +p, Act_R)$, since the proof for $(Act_T, \varphi, -p, Act_R)$ is similar.

Proposition 5.2. *Given* $N = (Q, \Pi, \mu, Act, \rightarrow)$, $q \in Q$ *and norm* $\mathsf{n} = (Act_T, \varphi, +p, Act_R) \in \mathcal{N}_2$, *the pointed normative system* $(N, q)[\mathsf{n}]$ *is norm-aligned with* n.

Proof. We assume an arbitrary finite (possibly empty) sequence of actions $\alpha_1 \ldots \alpha_n \in Act^*$ and proposition p'. We have:

$$(N, q)[\mathsf{n}] \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p' \quad \Leftrightarrow \quad N[\mathsf{n}], q^r \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p'$$

We consider either:

$$\exists i \in \{1, \ldots, n\} : (\alpha_i \in Act_T \text{ and } \forall j \in \{i+1, \ldots, n\} : \alpha_j \notin Act_R)$$

Or:

$$\forall i \in \{1, \ldots, n\} : (\alpha_i \in Act_T \Rightarrow \exists j \in \{i+1, \ldots, n\} : \alpha_j \in Act_R)$$

- In the first case, let α_i be the action for which $\alpha_i \in Act_T$ and $\forall j \in \{i+1, \ldots, n\}$ such that $\alpha_j \notin Act_R$. We have:

$$N[\mathsf{n}], q^r \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p' \quad \Leftrightarrow \quad N[\mathsf{n}], q^r(\alpha_1 \ldots \alpha_i) \vDash \Box_{\alpha_{i+1}} \cdots \Box_{\alpha_n} p' \quad \Leftrightarrow$$
$$N[\mathsf{n}], q(\alpha_1 \ldots \alpha_i)^a \vDash \Box_{\alpha_{i+1}} \cdots \Box_{\alpha_n} p' \quad \Leftrightarrow$$
$$N[\mathsf{n}], q(\alpha_1 \ldots \alpha_i)^a(\alpha_{i+1} \ldots \alpha_n) \vDash p' \quad \Leftrightarrow \quad N[\mathsf{n}], q(\alpha_1 \ldots \alpha_n)^a \vDash p' \quad \Leftrightarrow$$
$$N, q(\alpha_1 \ldots \alpha_n) \vDash p' \text{ or } (p' = p \text{ and } N, q(\alpha_1 \ldots \alpha_n) \vDash \varphi) \quad \Leftrightarrow$$
$$N, q \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p' \text{ or } (p' = p \text{ and } N, q \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} \varphi)$$

- In the second case, we either have that $\exists i \in \{1, \ldots, n\}$ such that $\alpha_i \in Act_T$ or we do not. If not, the following is obvious:

$$N[\mathsf{n}], q^r \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p' \quad \Leftrightarrow \quad N[\mathsf{n}], q(\alpha_1 \ldots \alpha_n)^r \vDash p' \quad \Leftrightarrow$$
$$N, q(\alpha_1 \ldots \alpha_n) \vDash p' \quad \Leftrightarrow \quad N, q \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p'$$

  If so, let α_i be the last occurring action in the sequence $\alpha_1 \ldots \alpha_n$ for which $\alpha_i \in Act_T$. We know that $i < n$, since there exists a $j \in \{i+1, \ldots, n\}$ for which $\alpha_j \in Act_R$. This implies that for all $j \in \{i+1, \ldots, n\}$ we have that $\alpha_j \in Act_R \backslash Act_T$, since α_i was the last occurring action for which $\alpha_i \in Act_T$. This implies:

$$N[\mathsf{n}], q^r \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p' \quad \Leftrightarrow \quad N[\mathsf{n}], q^r(\alpha_1 \ldots \alpha_i) \vDash \Box_{\alpha_{i+1}} \cdots \Box_{\alpha_n} p' \quad \Leftrightarrow$$
$$N[\mathsf{n}], q(\alpha_1 \ldots \alpha_i)^r \vDash \Box_{\alpha_{i+1}} \cdots \Box_{\alpha_n} p' \quad \Leftrightarrow$$
$$N[\mathsf{n}], q(\alpha_1 \ldots \alpha_i)^r(\alpha_{i+1} \ldots \alpha_n) \vDash p' \quad \Leftrightarrow$$
$$N[\mathsf{n}], q(\alpha_1 \ldots \alpha_n)^r \vDash p' \quad \Leftrightarrow \quad N, q(\alpha_1 \ldots \alpha_n) \vDash p' \quad \Leftrightarrow \quad N, q \vDash \Box_{\alpha_1} \cdots \Box_{\alpha_n} p'$$

□

We again return to our running example of the library to visualize how this new update works.

Running Example

We return to the running example of the library visualized in Figure 5.1. We now want to update this system with the norm which states that the lending of a magazine is forbidden for the customer. We do this by adding the norm $\mathsf{n}_1 = (\{\text{lend\_mag}\}, \top, +v, \{\text{return}\})$, which states that there is a violation v if 'lend_mag' has been performed, and this violation remains for all states until 'return' is performed (here, the action 'return' is seen as either the returning of a book or a magazine by the customer, or both). By following the rules of the update, we see in Figure 5.3 how we go from system N_{library} to $N_{\text{library}}[\mathsf{n}_1]$, and states q_0 and q_1 to q_0^r and q_1^r respectively. We have:

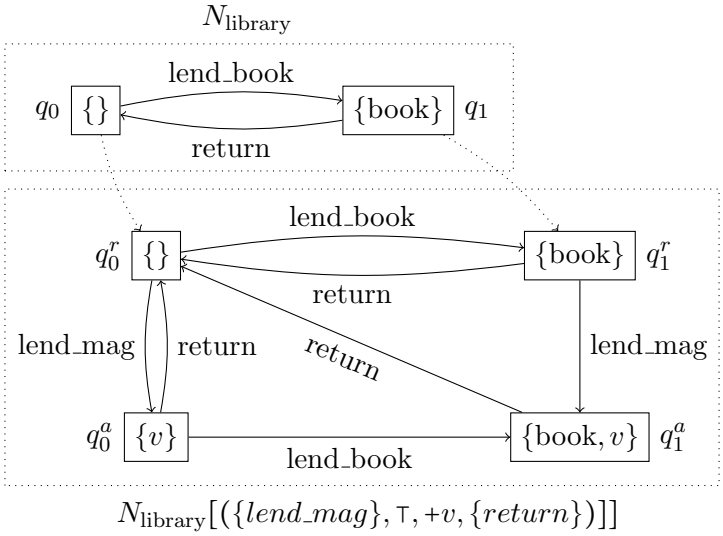


Figure 5.3: On top we have normative system $N_{library}$, and on the bottom we have normative system $N_{library}[(\{lend\_mag\}, \top, +mag, \{return\})]$. Again, the dotted transitions denote the state-updates that occur.

$$(N_{\text{library}}, q_0)[\mathsf{n}_1] \vDash \Box_{\text{lend\_mag}}(v)$$

To illustrate that norms from $\mathcal{N}_1$ and $\mathcal{N}_2$ can be combined, we can imagine that an additional norm is added which states that having a book removes the violation, i.e., the lending of a magazine is forbidden, unless the customer also has a book. This can be modelled by additionally adding the norm $\mathsf{n}_2 = (book, -v, \varnothing) \in \mathcal{N}_1$ to the updated system. Adding this norm implies that the lending of a magazine only causes a violation if the customer does not have a book. This is shown by the following formula:

$$((N_{\text{library}}, q_0)[\mathsf{n}_1])[\mathsf{n}_2] \vDash \Box_{\text{lend\_mag}}(v \wedge \Box_{\text{lend\_book}} \neg v)$$

In the next section we will extend our logical language even further to allow us to reason about these new dynamic updates.

5.4.2 Syntax and semantics

In this section, we extend language $\mathcal{L}_1$ again to reason about these new updates. Given a normative system $N = (Q, \Pi, \mu, Act, \rightarrow)$, the language of propositional logic with action modality including norm updates from norm language $\mathcal{N}_1$ and $\mathcal{N}_2$, written in this chapter as $\mathcal{L}_2$, consists of formulas φ built

by the following grammar, where $p \in \Pi$, $\alpha \in Act$, and $\mathsf{n} \in (\mathcal{N}_1 \cup \mathcal{N}_2)$:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \Box_\alpha \varphi \mid [\mathsf{n}]\varphi$$

From this grammar it is again apparent that $\mathcal{L}_0 \subset \mathcal{L}_1 \subset \mathcal{L}_2$. Along a pointed normative system (N, q), we can evaluate formulas of $\mathcal{L}_2$ as follows.

Definition 5.7 ($\mathcal{L}_2$ semantics)**.** *Given a normative system $N = (Q, \Pi, \mu, Act, \rightarrow)$ and a state $q \in Q$, the satisfaction relation $\vDash_{L_2}$ of $\mathcal{L}_2$ is inductively defined as follows:*

- $N, q \vDash_{L_2} p$ *iff* $p \in \mu(q)$
- $N, q \vDash_{L_2} \neg\varphi$ *iff* $N, q \nvDash_{L_2} \varphi$
- $N, q \vDash_{L_2} \varphi_1 \vee \varphi_2$ *iff* $N, q \vDash_{L_2} \varphi_1$ *or* $N, q \vDash_{L_2} \varphi_2$.
- $N, q \vDash_{L_2} \Box_\alpha \varphi$ *iff* $N, q(\alpha) \vDash_{L_2} \varphi$
- $N, q \vDash_{L_2} [\mathsf{n}]\varphi$ *iff* $(N, q)[\mathsf{n}] \vDash_{L_2} \varphi$

We again have that the satisfaction relation $\vDash_{L_2}$ *extends* the satisfaction relation $\vDash_{L_1}$, allowing us to simply write $\vDash$ instead of $\vDash_{L_2}$, $\vDash_{L_1}$, or $\vDash_{L_0}$, since there should be no confusion between the intended satisfaction relation. Let us look at some examples before we move to an axiomatization.

Examples

Before we go to an axiomatization of this logic, let us look at some specific (non-)validities of this extended dynamic logic. First, we expect that updating with $(Act_T, \varphi, +p, Act_R)$ does not immediately change the valuation of the current state. This is reflected by the following validity, which holds for any proposition p':

$$\vDash ([(Act_T, \varphi, +p, Act_R)]p') \leftrightarrow p'$$

In other words, the truth of p' in the current state after an update depends merely on the truth of p' before the update. Moreover, we expect whenever a norm effect is not triggered by a certain action, it does not matter whether this update is performed before or after this action. That is, whenever $\alpha_1, \ldots, \alpha_n \notin Act_T$, we have that:

$$\begin{aligned} \vDash\ & ([(Act_T, \varphi, +p, Act_R)](\Box_{\alpha_1} \cdots \Box_{\alpha_n} \psi)) \leftrightarrow \\ & (\Box_{\alpha_1} \cdots \Box_{\alpha_n} [(Act_T, \varphi, +p, Act_R)]\psi) \end{aligned}$$

Of course, a more interesting scenario happens when a trigger action is performed. If we have that $\alpha \in Act_T$, we have the following validity, which elegantly showcases the connection between updates from $\mathcal{N}_1$ and $\mathcal{N}_2$:

$$\begin{aligned} \vDash\ & ([(Act_T, \varphi, +p, Act_R)]\, \Box_\alpha\, \psi) \leftrightarrow \\ & (\Box_\alpha [\varphi, +p, Act_T \cup Act_R]([Act_T, \varphi, +p, Act_R]\psi)) \end{aligned}$$

The truth of this validity is not at all apparent, and a large portion of the next section will be devoted to proving this. In words, updating a system with $(Act_T, \varphi, +p, Act_R)$ and then performing a trigger action is equivalent to first performing the trigger action, then updating the system with $(\varphi, +p, Act_T \cup Act_R)$ (note that this norm is in $\mathcal{N}_1$, and that the repair actions are $Act_T \cup Act_R$ instead of just Act_R) and finally updating again with $(Act_T, \varphi, +p, Act_R)$. The reason the norm $(Act_T, \varphi, +p, Act_R)$ does not disappear on the right hand side is because it remains in effect, i.e. it can be triggered by an action at later stage.

In the next section we will provide an axiomatization of our extended dynamic logic which, as we will show later, will be both sound and complete.

5.4.3 Axiomatization

In the previous section, we saw some example validities which we can use in this section to provide an axiomatization of our logic. At the end of that section, we saw a validity which showed a correspondence between norms from $\mathcal{N}_1$ and from $\mathcal{N}_2$, which will play an important role here. Before we do, we need to introduce an extra concept.

Definition 5.8 (Triggered norms)**.** *Given a norm* $\mathsf{n} = (Act_T, \varphi, \pm p, Act_R) \in \mathcal{N}_2$*, we define* $\mathsf{n}_T \in \mathcal{N}_1$ *as follows:*

$$\mathsf{n}_T := (\varphi, \pm p, Act_T \cup Act_R)$$

We refer to n_T *as the* triggered norm*.*

The reason why n_T contains both the trigger and the repair actions on the right hand side will follow later from Lemma 5.6 (and Theorem 5.5), and in Proposition 5.4 we even establish that if we would define it in another way we would run into problems. For now, we will first focus on establishing an important bi-simulation result for $\mathcal{L}_2$. In Lemma 5.2 we established the result that if, for a given normative system N and N', there exists a bi-simulation Z over (N, N') for which $(q, q') \in Z$, then for any modal formula $\varphi \in \mathcal{L}_1$ we have:

$$N, q \vDash \varphi \quad \Leftrightarrow \quad N', q' \vDash \varphi$$

As we mentioned, an important consequence of this Lemma was that any formula from $\mathcal{L}_1$ is (equivalent to) a modal formula, which in turn led to our sound and complete axiomatization. In the upcoming Lemma, we also establish this result for $\mathcal{L}_2$.

Lemma 5.5. *Let N and N' be two normative systems, and let Z be a bi-simulation over (N, N'). Given that $(q, q') \in Z$, we have for every formula $\varphi \in \mathcal{L}_2$:*

$$N, q \vDash \varphi \quad \Leftrightarrow \quad N', q' \vDash \varphi$$

Proof. The proof is very similar to the proof found in Lemma 5.2. Given such a bi-simulation Z, let $Z[\mathsf{n}]$ again be defined as follows:

$$Z[\mathsf{n}] := \{(q_i^a, q_j^a), (q_i^r, q_j^r) \mid (q_i, q_j) \in Z\}$$

The important difference now is the observation that if Z is a bi-simulation over (N, N'), we have that $Z[\mathsf{n}]$ is a bi-simulation over $(N[\mathsf{n}], N'[\mathsf{n}])$ if $\mathsf{n} \in \mathcal{N}_1$ *or* if $\mathsf{n} \in \mathcal{N}_2$. We omit the proof of this claim, since it is only a small extension of the proof found in Lemma 5.1. Once we establish this, we can show in the same manner that if $(q_i, q_j) \in Z$, then they have to agree on all formulas from $\mathcal{L}_2$, i.e. all modal formulas including dynamic nested norm updates from *both* $\mathcal{N}_1$ and $\mathcal{N}_2$. We can prove this by induction on φ. The interesting case is when, from this assumption, we wish to conclude that this implies that they also both agree on $[\mathsf{n}]\varphi$, where $\mathsf{n} \in \mathcal{N}_2$. Assume $N, q_i \vDash [\mathsf{n}]\varphi$, from which we wish to show that this implies $N, q_j \vDash [\mathsf{n}]\varphi$. By our assumption that $N, q_i \vDash [\mathsf{n}]\varphi$, we have that $N[\mathsf{n}], q_i^a \vDash \varphi$. But, since $(q_i^a, q_j^a) \in Z[\mathsf{n}]$ where $Z[\mathsf{n}]$ is a bi-simulation over $(N[\mathsf{n}], N'[\mathsf{n}])$, we can use our induction hypothesis to conclude that $N'[\mathsf{n}], q_j^a \vDash \varphi$, and thus that $N, q_j \vDash [\mathsf{n}]\varphi$ as needed. The other direction is similar. □

Using this result, we can prove the following Lemma, which will play a crucial role in our axiomatization.

Lemma 5.6. *Given an arbitrary pointed normative system (N, q), norm $\mathsf{n} = (Act_T, \varphi, \pm p, Act_R) \in \mathcal{N}_2$ and formula $\psi \in \mathcal{L}_2$. We have:*

$$N[\mathsf{n}], q^a \vDash \psi \quad \Leftrightarrow \quad (N[\mathsf{n}_T], q^a)[\mathsf{n}] \vDash \psi$$

Proof. We define the following normative systems:

$$\begin{aligned} N_1 = N[\mathsf{n}] &= (Q_1, \Pi, \mu_1, Act, \to_1) \\ N_2 = N[\mathsf{n}_T] &= (Q_2, \Pi, \mu_2, Act, \to_2) \\ N_3 = N[\mathsf{n}_T][\mathsf{n}] &= (Q_3, \Pi, \mu_3, Act, \to_3) \end{aligned}$$

Since $q^a[\mathsf{n}] = (q^a)^r$, we have to show:

$$N_1, q^a \vDash \psi \quad \Leftrightarrow \quad N_3, (q^a)^r \vDash \psi$$

Define Z as follows:

$$Z = \{(q^a, (q^r)^a), (q^a, (q^a)^r), (q^r, (q^r)^r) \mid q \in Q\}$$

Note that for a given state $q \in Q$ we have that $(q^a, (q^a)^a) \notin Z$, which is of crucial importance to show that the first bi-simulation condition indeed holds. To show the first condition, assume an arbitrary state $q \in Q$. We have to show that $\mu_1(q^a) = \mu_3((q^r)^a)$, $\mu_1(q^a) = \mu_3((q^a)^r)$, and, $\mu_1(q^r) = \mu_3((q^r)^r)$, which we will do separately:

1. $\mu_1(q^a) = \mu_3((q^r)^a)$. Assume that $N, q \vDash \varphi$. In this case, we have by definition that $\mu_1(q^a) = \mu(q) \cup \{p\}$. Moreover, by Lemma 5.3 we have that this implies that $N_2, q^r \vDash \varphi$, which in turn implies that $\mu_3((q^r)^a) = \mu_2(q^r) \cup \{p\} = \mu(q) \cup \{p\}$. This allows us to conclude that $\mu_1(q^a) = \mu_3((q^r)^a)$, as needed. Now assume that $N, q \nvDash \varphi$. In this case, we have by definition that $\mu_1(q^a) = \mu(q)$. Moreover, by Lemma 5.3 we have that this implies that $N_2, q^r \nvDash \varphi$, which implies that $\mu_3((q^r)^a) = \mu_2(q^r) = \mu(q)$. This again allows us to conclude that $\mu_1(q^a) = \mu_3((q^r)^a)$, as needed.

2. $\mu_1(q^a) = \mu_3((q^a)^r)$. Assume that $N, q \vDash \varphi$. In this case, we have by definition that $\mu_1(q^a) = \mu(q) \cup \{p\}$. Moreover, by definition we have that $\mu_3((q^a)^r) = \mu_2(q^a) = \mu(q) \cup \{p\}$. This allows us to conclude that $\mu_1(q^a) = \mu_3((q^r)^a)$, as needed. Now assume that $N, q \nvDash \varphi$. In this case, we have by definition that $\mu_1(q^a) = \mu(q)$. Moreover, by definition we have that $\mu_3((q^a)^r) = \mu_2(q^a) = \mu(q)$. This again allows us to conclude that $\mu_1(q^a) = \mu_3((q^r)^a)$, as needed.

3. $\mu_1(q^r) = \mu_3((q^r)^r)$. This one is trivial, since by definition $\mu_1(q^r) = \mu(q) = \mu_2(q^r) = \mu_3((q^r)^r)$.

This concludes the first bi-simulation condition. For the second condition, assume an arbitrary $(q^x, (q^y)^z) \in Z$ (where x, y or z can be either r or a). We consider an arbitrary action $\alpha \in Act$, for which it is either the case that $q^x(\alpha) = q(\alpha)^a$ or $q^x(\alpha) = q(\alpha)^r$:

- If $q^x(\alpha) = q(\alpha)^a$, it follows that $x = r$ and $\alpha \in Act_T$, or $x = a$ and $\alpha \notin Act_R \backslash Act_T$. If $x = r$ and $\alpha \in Act_T$, we have that $(q^y)^z(\alpha) = (q(\alpha)^r)^a$ for which we have $(q(\alpha)^a, (q(\alpha)^r)^a) \in Z$. If $x = a$ and $\alpha \notin Act_R \backslash Act_T$, we have that $(q^y)^z(\alpha) = (q(\alpha)^a)^r$ or $(q^y)^z(\alpha) = (q(\alpha)^r)^a$, for which $(q(\alpha)^a, (q(\alpha)^r)^a) \in Z$ and $(q(\alpha)^a, (q(\alpha)^a)^r) \in Z$.

Axiom schemata	
2.1.	$([\mathsf{n}]p') \leftrightarrow p'$
2.2.	$([\mathsf{n}]\neg\psi) \leftrightarrow (\neg[\mathsf{n}]\psi)$
2.3.	$([\mathsf{n}](\psi_1 \vee \psi_2)) \leftrightarrow (([\mathsf{n}]\psi_1) \vee ([\mathsf{n}]\psi_2))$
2.4.	$([\mathsf{n}]\,\Box_\alpha\,\psi) \leftrightarrow (\Box_\alpha[\mathsf{n}_T]([\mathsf{n}]\psi))$ (if $\alpha \in Act_T$)
2.5.	$([\mathsf{n}]\,\Box_\alpha\,\psi) \leftrightarrow (\Box_\alpha[\mathsf{n}]\psi)$ (if $\alpha \notin Act_T$)
+	All axioms and rules from system **L1**, including the rule for replacement of equivalent formulas (RE).

Table 5.2: System **L2***, where* $\mathsf{n} = (Act_T, \varphi, \pm p, Act_R) \in \mathcal{N}_2$*. System* **L2** *extends system* **L1** *by adding these five additional axiom schemes.*

- If $q^x(\alpha) = q(\alpha)^r$, it follows that $x = a$ and $\alpha \in Act_R \backslash Act_T$ or $x = r$ and $\alpha \notin Act_T$. In both cases, we have that $(q^y)^z(\alpha) = (q(\alpha)^r)^r$ for which we have $(q(\alpha)^r, (q(\alpha)^r)^r) \in Z$.

From these cases we can conclude that for every $q \in Q$ and $\alpha \in Act$ that $(q^x(\alpha), (q^y)^z(\alpha)) \in Z$, which shows the second bi-simulation condition. The final part of this proof is the observation that for initial state q we have $(q^a, (q^a)^r) \in Z$, and thus for every $\psi \in \mathcal{L}_2$:

$$N_1, q^a \vDash \psi \quad \Leftrightarrow \quad N_3, (q^a)^r \vDash \psi$$

As needed. □

In words, given a norm $\mathsf{n} \in \mathcal{N}_2$, this lemma states that evaluating any formula in an updated system $N[\mathsf{n}]$ at an active state, is equivalent to evaluating a formula in the system updated with the triggered norm $N[\mathsf{n}_T]$ at the same state, making sure that we also update the latter system and state with the original norm n.

We are now ready to give the axiomatization of $\mathcal{L}_2$, which is given by System **L2** shown in Table 5.2. We see that system **L2** extends system **L1** by five additional axiom schemes. We again write $\vdash_{\mathrm{L}_2} \varphi$ whenever a formula $\varphi \in \mathcal{L}_2$ is provable from the rules and axioms from **L2**. Axiom 2.1 states that an update does not modify the state of affairs of the current state, i.e. an arbitrary proposition is true in the current state after the update if it was true before the update. Axiom 2.2 and 2.3 both state that the dynamic update is a *function*: given a normative system and an update, a new normative system is uniquely determined. They are similar to axioms 1.4 and 1.5, except they apply to norms from $\mathcal{N}_2$ instead of $\mathcal{N}_1$. Axiom 2.4 is the interesting one, and states that when a trigger action has been performed, it would be equivalent to first performing this action, then adding the triggered norm, and finally

adding the norm again. This axiom showcases the connection between our two classes of norms, since again we have that for a given norm $\mathsf{n} \in \mathcal{N}_2$ that $\mathsf{n}_T \in \mathcal{N}_1$. The order in which these updates occur, and the way that n_T is defined, is of crucial importance for the soundness of this scheme. After we show soundness of these rules in Theorem 5.5, we will show in Proposition 5.3 and Proposition 5.4 that if we would change this axiom even slightly, soundness will disappear. Axiom 2.5 states the property that if the norm is not triggered by a certain action, it does not matter if we perform the update before or after this action. Note that these axioms are only concerned with the trigger actions, while the axioms which are part of **L1** are only concerned with the repair actions. Together they form a complete axiomatization which considers both the trigger and repair actions. We have the following important result.

Theorem 5.5 (Soundness of **L2**)**.** *System* **L2***, as shown in Table 5.2, is sound with respect to the corresponding semantics of logic* $\mathcal{L}_2$*. That is, for every formula* $\varphi \in \mathcal{L}_2$ *we have:*

$$\vdash_{L_2} \varphi \quad \Rightarrow \quad \vDash_{L_2} \varphi$$

Proof. Soundness can be proven by proving soundness for every axiom independently. Furthermore, soundness of rule (RE) transfers to system **L2**, since the same proof from Lemma 5.4 applies in $\mathcal{L}_2$. Every step in this proof is some application of a definition found earlier in this chapter, except for axiom 2.4 where we use the important result acquired in Lemma 5.6 (denoted with $\overset{!}{\Leftrightarrow}$).

1.
$$\begin{array}{ccccc} N,q \vDash [\mathsf{n}]p' & \Leftrightarrow & N[\mathsf{n}],q^r \vDash p' & \Leftrightarrow \\ p' \in \mu'(q^r) & \Leftrightarrow & p' \in \mu(q) & \Leftrightarrow \\ \multicolumn{5}{c}{N,q \vDash p'} \end{array}$$

2.
$$\begin{array}{ccccc} N,q \vDash [\mathsf{n}]\neg\psi & \Leftrightarrow & (N,q)[\mathsf{n}] \vDash \neg\psi & \Leftrightarrow \\ (N,q)[\mathsf{n}] \nvDash \psi & \Leftrightarrow & N,q \nvDash [\mathsf{n}]\psi & \Leftrightarrow \\ \multicolumn{5}{c}{N,q \vDash \neg[\mathsf{n}]\psi} \end{array}$$

3.
$$\begin{array}{ccc} N,q \vDash [\mathsf{n}](\psi_1 \vee \psi_2) \quad \Leftrightarrow \quad (N,q)[\mathsf{n}] \vDash \psi_1 \vee \psi_2 & \Leftrightarrow \\ (N,q)[\mathsf{n}] \vDash \psi_1 \text{ or } (N,q)[\mathsf{n}] \vDash \psi_2 & \Leftrightarrow \\ N,q \vDash [\mathsf{n}]\psi_1 \text{ or } N,q \vDash [\mathsf{n}]\psi_2 & \Leftrightarrow \\ N,q \vDash ([\mathsf{n}]\psi_1) \vee ([\mathsf{n}]\psi_2) \end{array}$$

4.

$$\begin{array}{rl}
N, q \vDash [\mathsf{n}]\, \Box_\alpha\, \psi & \Leftrightarrow \\
N[\mathsf{n}], q^r \vDash \Box_\alpha \psi & \Leftrightarrow \\
 & \overset{!}{\Leftrightarrow} \\
N[\mathsf{n}], q(\alpha)^a \vDash \psi \quad (\text{if } \alpha \in Act_T) & \Leftrightarrow \\
(N[\mathsf{n}_T], q(\alpha)^a)[\mathsf{n}] \vDash \psi \quad (\text{if } \alpha \in Act_T) & \Leftrightarrow \\
N[\mathsf{n}_T], q(\alpha)^a \vDash [\mathsf{n}]\psi \quad (\text{if } \alpha \in Act_T) & \Leftrightarrow \\
N, q(\alpha) \vDash [\mathsf{n}_T]([\mathsf{n}]\psi) \quad (\text{if } \alpha \in Act_T) & \Leftrightarrow \\
N, q \vDash \Box_\alpha [\mathsf{n}_T]([\mathsf{n}]\psi) \quad (\text{if } \alpha \in Act_T) &
\end{array}$$

5.

$$\begin{array}{rl}
N, q \vDash [\mathsf{n}]\, \Box_\alpha\, \psi & \Leftrightarrow \\
N[\mathsf{n}], q^r \vDash \Box_\alpha \psi & \Leftrightarrow \\
N[\mathsf{n}], q^r(\alpha) \vDash \psi & \Leftrightarrow \\
N[\mathsf{n}], q(\alpha)^r \vDash \psi \ (\text{if } \alpha \notin Act_T) & \Leftrightarrow \\
N, q(\alpha) \vDash [\mathsf{n}]\psi \ (\text{if } \alpha \notin Act_T) & \Leftrightarrow \\
N, q \vDash \Box_\alpha [\mathsf{n}]\psi \ (\text{if } \alpha \notin Act_T) &
\end{array}$$

□

Since axiom 2.4 is so important, we will show that slight variations of this axiom are **not** sound. An example of a variation might be to change the order in which $[\mathsf{n}]$ and $[\mathsf{n}_T]$ appear on the right side of this axiom. We have the following result.

Proposition 5.3. *The axiom scheme:*

$$([\mathsf{n}]\, \Box_\alpha\, \psi) \leftrightarrow (\Box_\alpha [\mathsf{n}]([\mathsf{n}_T]\psi)) \ (\textit{if } \alpha \in Act_T)$$

Is **not** *sound.*

Proof. We show this by showing that there exists a pointed normative system (N, q), norm $(Act_T, \varphi, \pm p, Act_R) \in \mathcal{N}_2$, action $\alpha \in Act_T$ and formula $\psi \in \mathcal{L}_2$ such that $N, q \nvDash [\mathsf{n}]\, \Box_\alpha\, \psi$ and $N, q \vDash \Box_\alpha [\mathsf{n}_{T'}]([\mathsf{n}]\psi)$. Consider the normative system N consisting of a single state $Q = \{q\}$, soft facts $\Pi_{\text{soft}} = \{p\}$, valuation function $\mu(q) = \varnothing$, and a single action $Act = \{\alpha\}$. Now consider the norm $(\{\alpha\}, \Box_\alpha \neg p, +p, \varnothing) \in \mathcal{N}_2$, which implies that $\mathsf{n}_T = (\Box_\alpha \neg p, +p, \{\alpha\}) \in \mathcal{N}_1$. Norm n states that after the performance of action α, for all states for which after the performance of α it was the case that $\neg p$ holds, p will start to hold. Moreover, let $\psi = p$. We can now show that:

$$N, q \vDash [\mathsf{n}]\, \Box_\alpha\, \psi \quad \text{and} \quad N, q \nvDash \Box_\alpha [\mathsf{n}][\mathsf{n}_T]\psi$$

Which is sufficient to prove that the axiom scheme is not sound. To show that $N, q \vDash [\mathsf{n}]\,\Box_\alpha\,\psi$, we observe the following equivalences:

$$\begin{aligned} N, q \vDash [\mathsf{n}]\,\Box_\alpha\, p \quad &\Leftrightarrow \quad N[\mathsf{n}], q^r \vDash \Box_\alpha p \quad \Leftrightarrow \\ N[\mathsf{n}], q^r(\alpha) \vDash p \quad &\Leftrightarrow \quad N[\mathsf{n}], q^a \vDash p \quad \Leftrightarrow \\ N, q \vDash \Box_\alpha \neg p \quad &\Leftrightarrow \quad N, q \vDash \neg p \end{aligned}$$

Since $N, q \vDash \neg p$, we can conclude that $N, q \vDash [\mathsf{n}]\,\Box_\alpha\, p$. In order to show that it holds that $N, q \nvDash \Box_\alpha[\mathsf{n}][\mathsf{n}_T]\varphi$, we observe the following equivalences:

$$\begin{aligned} N, q \vDash \Box_\alpha[\mathsf{n}][\mathsf{n}_T]p \quad &\Leftrightarrow \quad N, q \vDash [\mathsf{n}][\mathsf{n}_T]p \quad \Leftrightarrow \\ N[\mathsf{n}], q^r \vDash [\mathsf{n}_T]p \quad &\Leftrightarrow \quad (N[\mathsf{n}])[\mathsf{n}_T], (q^r)^a \vDash p \quad \Leftrightarrow \\ N[\mathsf{n}], q^r \vDash p \text{ or } N[\mathsf{n}], q^r \vDash \Box_\alpha \neg p \quad &\Leftrightarrow \quad N, q \vDash p \text{ or } N[\mathsf{n}], q^r \vDash \Box_\alpha \neg p \quad \Leftrightarrow \\ N[\mathsf{n}], q^r \vDash \Box_\alpha \neg p \quad &\Leftrightarrow \quad N[\mathsf{n}], q^r(\alpha) \vDash \neg p \quad \Leftrightarrow \\ N[\mathsf{n}], q^a \nvDash p \quad &\Leftrightarrow \quad N, q \nvDash p \text{ and } N, q \nvDash \Box_\alpha \neg p \quad \Leftrightarrow \\ N, q \nvDash p \text{ and } N, q(\alpha) \nvDash \neg p \quad &\Leftrightarrow \quad N, q \nvDash p \text{ and } N, q \vDash p \end{aligned}$$

Since the latter ($N, q \nvDash p$ and $N, q \vDash p$) is trivially false, we can conclude that $N, q \nvDash \Box_\alpha[\mathsf{n}][\mathsf{n}_T]p$, as needed. □

The 'problem' that arises if we update in the wrong order, the norm n and n_T start to 'interfere' with each other. That is, the norm condition and effect of n influences the norm condition and effect of n_T. By reversing the order in which these updates occur, the interference disappears. This interference also occurs if n_T would be defined in a slightly different manner. For example, consider the following definition:

$$(Act_T, \varphi, \pm p, Act_R)_{T'} := (\varphi, \pm p, Act_R)$$

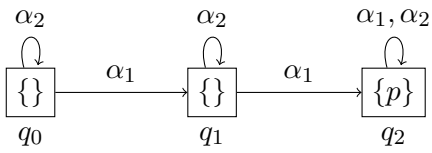
Notice that now, given a norm $\mathsf{n} \in \mathcal{N}_2$, we have only Act_R as the repair actions of $\mathsf{n}_{T'}$, instead of $Act_T \cup Act_R$ in the case of n_T. Whenever for a given norm $\mathsf{n} = (Act_T, \varphi, \pm p, Act_R)$ we have that $Act_T \subseteq Act_R$, we have that these definitions coincide, i.e. $\mathsf{n}_T = \mathsf{n}_{T'}$, but whenever this is not the case we have that $\mathsf{n}_T \neq \mathsf{n}_{T'}$. This definition of a triggered norm might initially appear to be the correct one to use in axiom 2.5, since it is perhaps not immediately apparent why the trigger actions should contribute to the repair of the norm. However, we have the following result.

Proposition 5.4. *The axiom scheme:*

$$([\mathsf{n}]\,\Box_\alpha\,\psi) \leftrightarrow (\Box_\alpha[\mathsf{n}_{T'}]([\mathsf{n}]\psi))\ (\textit{if } \alpha \in Act_T)$$

Is **not** *sound.*

Proof. We show this by showing that there exists a pointed normative system (N, q), norm $(Act_T, \varphi, \pm p, Act_R) \in \mathcal{N}_2$, action $\alpha \in Act_T$ and formula $\psi \in \mathcal{L}_2$ such that $N, q \not\models [\mathsf{n}]\, \Box_\alpha\, \psi$ and $N, q \models \Box_\alpha [\mathsf{n}_{T'}]([\mathsf{n}]\psi)$. Consider the following normative system N:



We have $Q = \{q_0, q_1, q_2\}$, $\Pi_{\text{soft}} = \{p\}$, and $Act = \{\alpha_1, \alpha_2\}$. We now consider the following norm $\mathsf{n} = (\{\alpha_2\}, \Box_{\alpha_1} p, +p, \varnothing) \in \mathcal{N}_2$. Note that $\mathsf{n}_T \neq \mathsf{n}_{T'}$, since $\mathsf{n}_T = (\Box_{\alpha_1} p, +p, \{\alpha_2\})$ and $\mathsf{n}_{T'} = (\Box_{\alpha_1} p, +p, \varnothing)$. Moreover, let $\psi = (\Box_{\alpha_2} p) \in \mathcal{L}_2$. We can now show that:

$$N, q_0 \not\models [\mathsf{n}]\, \Box_{\alpha_2}\, \psi \quad \text{and} \quad N, q_0 \models \Box_{\alpha_2} [\mathsf{n}_{T'}][\mathsf{n}]\psi$$

Which is sufficient to prove that the axiom scheme is not sound. To show that $N, q_0 \not\models [\mathsf{n}]\, \Box_{\alpha_2}\, \psi$, we observe the following equivalences:

$$\begin{aligned}
N, q_0 \models [\mathsf{n}]\, \Box_{\alpha_2}\, \Box_{\alpha_2} p \quad &\Leftrightarrow \quad N[\mathsf{n}], q_0^r \models \Box_{\alpha_2}\, \Box_{\alpha_2}\, p \quad \Leftrightarrow \\
N[\mathsf{n}], q_0^r(\alpha_2\alpha_2) \models p \quad &\Leftrightarrow \quad N[\mathsf{n}], q_0^a \models p \quad \Leftrightarrow \\
N, q_0 \models \Box_{\alpha_1} p \quad &\Leftrightarrow \quad N, q_1 \models p
\end{aligned}$$

However, since $N, q_1 \not\models p$, we can conclude that $N, q_0 \not\models [\mathsf{n}]\, \Box_{\alpha_2}\, \psi$. To show that $N, q_0 \models \Box_{\alpha_2} [\mathsf{n}_{T'}][\mathsf{n}]\psi$, we again observe the following equivalences:

$$\begin{aligned}
N, q_0 \models \Box_{\alpha_2} [\mathsf{n}_{T'}][\mathsf{n}]\, \Box_{\alpha_2}\, p \quad &\Leftrightarrow \quad N, q_0 \models [\mathsf{n}_{T'}][\mathsf{n}]\, \Box_{\alpha_2}\, p \quad \Leftrightarrow \\
N[\mathsf{n}_{T'}], q_0^a \models [\mathsf{n}]\, \Box_{\alpha_2}\, p \quad &\Leftrightarrow \quad N[\mathsf{n}_{T'}][\mathsf{n}], (q_0^a)^r \models \Box_{\alpha_2} p \quad \Leftrightarrow \\
N[\mathsf{n}_{T'}][\mathsf{n}], (q_0^a)^r(\alpha_2) \models p \quad &\overset{!}{\Leftrightarrow} \quad N[\mathsf{n}_{T'}][\mathsf{n}], (q_0^a)^a \models p \quad \Leftrightarrow \\
N[\mathsf{n}_{T'}], q_0^a \models p \text{ or } N[\mathsf{n}_{T'}], q_0^a \models \Box_{\alpha_1} p \quad &\Leftrightarrow \quad N[\mathsf{n}_{T'}], q_0^a \models \Box_{\alpha_1} p \quad \Leftrightarrow \\
N[\mathsf{n}_{T'}], q_1^a \models p \quad &\Leftrightarrow \quad N, q_1 \models p \text{ or } N, q_1 \models \Box_{\alpha_1} p \quad \Leftrightarrow \\
N, q_1 \models \Box_{\alpha_1} p \quad &\Leftrightarrow \quad N, q_2 \models p
\end{aligned}$$

Since $N, q_2 \models p$, we can conclude that $N, q_0 \models \Box_{\alpha_2} [\mathsf{n}_{T'}][\mathsf{n}]\psi$. In the above, we have marked with $\overset{!}{\Leftrightarrow}$ the first occurrence which would be different if we would have used n_T instead of $\mathsf{n}_{T'}$, which is where the unsoundness of the axiom scheme arises. To illustrate this, in case of n_T, we would have:

$$N[\mathsf{n}_T][\mathsf{n}], (q_0^a)^r(\alpha_2) \models p \quad \Leftrightarrow \quad N[\mathsf{n}_T][\mathsf{n}], (q_0^r)^a \models p$$

Where special notice should be laid on the difference between $(q_0^r)^a$ and $(q_0^a)^a$. □

We are now ready to show that system **L2** is complete. Again, we can show this by demonstrating that our rules and axioms allow for reduction. Particularly, these rules and axioms allow us to reduce formulas of $\mathcal{L}_2$ to formulas of $\mathcal{L}_0$, which allows us to show completeness. This result is shown in the next Theorem.

Theorem 5.6 (Completeness of **L2**). *System* **L2**, *as shown in Table 5.2, is complete with respect to the corresponding semantics of logic* $\mathcal{L}_2$. *That is, for every formula* $\varphi \in \mathcal{L}_2$ *we have:*

$$\vDash_{L_2} \varphi \quad \Rightarrow \quad \vdash_{L_2} \varphi$$

Proof. In order to show completeness, we define the following translation function $\tau_2 : \mathcal{L}_2 \to \mathcal{L}_0$, where below the function $\tau_1 : \mathcal{L}_1 \to \mathcal{L}_0$ is the function as defined in the proof of Theorem 5.3, which we use to reduce formulas of the form $[\mathsf{n}]p$ (if $\mathsf{n} \in \mathcal{N}_1$):

$$\begin{array}{rcll}
\tau_2(p) & = & p & \\
\tau_2([\mathsf{n}]p) & = & \tau_1([\mathsf{n}]p) & (\text{if } \mathsf{n} \in \mathcal{N}_1) \\
\tau_2([\mathsf{n}]p) & = & p & (\text{if } \mathsf{n} \in \mathcal{N}_2) \\
\tau_2(\neg\psi) & = & \neg\tau_2(\psi) & \\
\tau_2(\psi_1 \vee \psi_2) & = & \tau_2(\psi_1) \vee \tau_2(\psi_2) & \\
\tau_2(\Box_\alpha\psi) & = & \Box_\alpha\tau_2(\psi) & \\
\tau_2([\mathsf{n}]\neg\psi) & = & \tau_2(\neg[\mathsf{n}]\psi) & \\
\tau_2([\mathsf{n}](\psi_1 \vee \psi_2)) & = & \tau_2(([\mathsf{n}]\psi_1) \vee ([\mathsf{n}]\psi_2)) & \\
\tau_2([\mathsf{n}]\,\Box_\alpha\,\psi) & = & \tau_2(\Box_\alpha\psi) & (\text{if } \mathsf{n} \in \mathcal{N}_1 \text{ and } \alpha \in Act_R) \\
\tau_2([\mathsf{n}]\,\Box_\alpha\,\psi) & = & \tau_2(\Box_\alpha[\mathsf{n}]\psi) & (\text{if } \mathsf{n} \in \mathcal{N}_1 \text{ and } \alpha \notin Act_R) \\
\tau_2([\mathsf{n}]\,\Box_\alpha\,\psi) & = & \tau_2(\Box_\alpha[\mathsf{n}_T]([\mathsf{n}]\psi)) & (\text{if } \mathsf{n} \in \mathcal{N}_2 \text{ and } \alpha \in Act_T) \\
\tau_2([\mathsf{n}]\,\Box_\alpha\,\psi) & = & \tau_2(\Box_\alpha[\mathsf{n}]\psi) & (\text{if } \mathsf{n} \in \mathcal{N}_2 \text{ and } \alpha \notin Act_T) \\
\tau_2([\mathsf{n}_1]([\mathsf{n}_2]\psi)) & = & \tau_2([\mathsf{n}_1]\tau_2([\mathsf{n}_2]\psi)) &
\end{array}$$

This time around, it is slightly harder to show that τ_2 is well-defined. This is because τ_2 on input $[\mathsf{n}]\Box_\alpha\psi$ for which $\mathsf{n} \in \mathcal{N}_2$ and $\alpha \in Act_T$ 'generates' an extra norm n_T, i.e., it can occur that on a certain input, the number of dynamic operators increases. However, since the number of norms from the set $\mathcal{N}_2$ in the scope of a norm from $\mathcal{N}_1$ never increases and always eventually decreases, we know that the translation will eventually return a well-formed formula from $\mathcal{L}_0$. The remaining steps in the proof follow the same pattern as the proof in Theorem 5.3. That is, we show in (1) that this translation has the property that for any $\varphi \in \mathcal{L}_2$ and $\mathsf{n} \in (\mathcal{N}_1 \cup \mathcal{N}_2)$ we have that $\vdash_{\mathrm{L}_2} [\mathsf{n}]\tau_2(\varphi) \leftrightarrow \tau_2([\mathsf{n}]\varphi)$. In (2), we show that from (1) we can conclude that for any $\varphi \in \mathcal{L}_1$ we have $\vdash_{\mathrm{L}_2} \varphi \leftrightarrow \tau_2(\varphi)$. In (3) we prove our completeness claim using the result from (2).

1. The first step in the completeness proof is to show that for any $\varphi \in \mathcal{L}_2$ and $\mathsf{n} \in (\mathcal{N}_1 \cup \mathcal{N}_2)$ that $\vdash_{\mathrm{L}_2} [\mathsf{n}]\tau_2(\varphi) \leftrightarrow \tau_2([\mathsf{n}]\varphi)$. We show this by induction on the length of φ. Let $\vdash_{\mathrm{L}_1} [\mathsf{n}]\tau_2(\varphi) \leftrightarrow \tau_2([\mathsf{n}]\varphi)$ be our induction hypotheses. We prove from this assumption that $\vdash_{\mathrm{L}_1} [\mathsf{n}]\tau(\Box_\alpha\varphi) \leftrightarrow \tau_1([\mathsf{n}](\Box_\alpha\varphi))$, and leave the other logical connectives (and the base cases) to the reader. We assume that $\mathsf{n} \in \mathcal{N}_2$, since the case where $\mathsf{n} \in \mathcal{N}_1$ is already shown in Theorem 5.3. We either have that $\alpha \in Act_T$ or not, and differentiate between these two cases.

   (a) If $\alpha \in Act_T$, we have due to axiom 2.4 and the definition of τ_2 that $\vdash_{\mathrm{L}_2} [\mathsf{n}](\tau_2(\Box_\alpha\varphi)) \leftrightarrow \Box_\alpha[\mathsf{n}_T]([\mathsf{n}]\tau_2(\varphi))$. By the rule of replacement of equivalent formulas (RE) from our induction hypothesis we get the result that $\vdash_{\mathrm{L}_2} [\mathsf{n}](\tau_2(\Box_\alpha\varphi)) \leftrightarrow \tau_2(\Box_\alpha[\mathsf{n}_T](\tau_2([\mathsf{n}]\varphi)))$. Since $\tau_2([\mathsf{n}]\varphi) \in \mathcal{L}_0$, it follows that the formula $([\mathsf{n}_T]\tau_2([\mathsf{n}]\varphi) \leftrightarrow \tau_2([\mathsf{n}_T](\tau_2([\mathsf{n}]\varphi)))) \in \mathcal{L}_1$. Moreover, since this formula is valid, we have due to completeness of $\mathcal{L}_1$ as shown in Theorem 5.3 that $\vdash_{\mathrm{L}_1} [\mathsf{n}_T]\tau_2([\mathsf{n}]\varphi) \leftrightarrow \tau_2([\mathsf{n}_T](\tau_2([\mathsf{n}]\varphi)))$, allowing us to conclude $\vdash_{\mathrm{L}_2} [\mathsf{n}_T]\tau_2([\mathsf{n}]\varphi) \leftrightarrow \tau_2([\mathsf{n}_T](\tau_2([\mathsf{n}]\varphi)))$ since **L2** contains all the rules and axioms of **L1**. Again, by the rule of replacement of equivalent formulas (RE) this allows us to conclude $\vdash_{\mathrm{L}_2} [\mathsf{n}](\tau_2(\Box_\alpha\varphi)) \leftrightarrow \Box_\alpha\tau_2([\mathsf{n}_T](\tau_2([\mathsf{n}]\varphi)))$. But by definition of τ_2 we have that:

$$\Box_\alpha\tau_2([\mathsf{n}_T](\tau_2([\mathsf{n}]\varphi))) = \Box_\alpha\tau_2([\mathsf{n}_T]([\mathsf{n}]\varphi)) = \tau_2(\Box_\alpha[\mathsf{n}_T]([\mathsf{n}]\varphi)) = \tau_2([\mathsf{n}]\,\Box_\alpha\,\varphi)$$

   We can conclude that $\vdash_{\mathrm{L}_2} [\mathsf{n}](\tau_2(\Box_\alpha\varphi)) \leftrightarrow \tau_2([\mathsf{n}]\,\Box_\alpha\,\varphi)$, as needed.

   (b) If $\alpha \notin Act_T$, we have due to axiom 2.5 that $\vdash_{\mathrm{L}_2} [\mathsf{n}](\tau_2(\Box_\alpha\varphi)) \leftrightarrow \Box_\alpha[\mathsf{n}]\tau_2(\varphi)$. By the rule of replacement of equivalent formulas (RE) from our induction hypothesis we get $\vdash_{\mathrm{L}_2} [\mathsf{n}](\tau_2(\Box_\alpha\varphi)) \leftrightarrow \Box_\alpha\tau_2([\mathsf{n}]\varphi)$, and by definition of τ_2 we get $\vdash_{\mathrm{L}_2} [\mathsf{n}](\tau_2(\Box_\alpha\varphi)) \leftrightarrow \tau_2([\mathsf{n}]\,\Box_\alpha\,\varphi)$, as needed.

2. The second step in the proof is to conclude that for any $\varphi \in \mathcal{L}_2$ we have that $\vdash_{\mathrm{L}_2} \varphi \leftrightarrow \tau_2(\varphi)$. We can again show this by induction on the length of φ. Let $\vdash_{\mathrm{L}_2} \varphi \leftrightarrow \tau_2(\varphi)$ be our induction hypothesis. The interesting case is when we want to show that this implies $\vdash_{\mathrm{L}_2} [\mathsf{n}]\varphi \leftrightarrow \tau_2([\mathsf{n}]\varphi)$. We already established that $\vdash_{\mathrm{L}_2} [\mathsf{n}]\tau_2(\varphi) \leftrightarrow \tau_2([\mathsf{n}]\varphi)$, allowing us to apply rule the rule of replacement of equivalent formulas (RE) from our induction hypothesis to achieve $\vdash_{\mathrm{L}_2} [\mathsf{n}]\varphi \leftrightarrow \tau_2([\mathsf{n}]\varphi)$. We leave the other logical connectives to the reader.

3. The third and final step is to show our completeness claim, i.e. from the assumption that $\vDash_{\mathrm{L}_2} \varphi$ we have $\vdash_{\mathrm{L}_2} \varphi$. By soundness of $\mathcal{L}_2$ (Theorem 5.5),

we have from $\vdash_{L_2} \varphi \leftrightarrow \tau_2(\varphi)$ that $\vDash_{L_2} \varphi \leftrightarrow \tau_2(\varphi)$. Thus, from our assumption that $\vDash_{L_2} \varphi$, we have $\vDash_{L_2} \tau_2(\varphi)$. By the completeness of system **L0** (Theorem 5.1), we have $\vdash_{L_0} \tau_2(\varphi)$. This implies that also $\vdash_{L_2} \tau_2(\varphi)$, since **L2** contains all the rules and axioms of **L0**. Since we have shown that $\vdash_{L_2} \varphi \leftrightarrow \tau_2(\varphi)$ and since $\vdash_{L_2} \tau_2(\varphi)$, we can apply rule MP to acquire $\vdash_{L_2} \varphi$.

□

This concludes our result on soundness and completeness of system **L2** for logic $\mathcal{L}_2$. In the above proof of the completeness of **L2**, we remarked that the function τ_2 was well-defined for any input $\varphi \in \mathcal{L}_2$. We reasoned that because in this translations norms from $\mathcal{N}_2$ in the scope of a norm from $\mathcal{N}_1$ never increases and always eventually decreases, termination of this translation process is guaranteed. To see that it is of crucial importance that norms from $\mathcal{N}_2$ in the scope of a norm from $\mathcal{N}_1$ never increases, we have the following proposition which shows that if this would not be the case in our translation, we run into problems.

Proposition 5.5. *Let τ_2' be defined exactly the same as τ_2 from Theorem 5.6, except where we replace:*

$$\tau_2([\mathsf{n}]\,\Box_\alpha\,\psi) = \tau_2(\Box_\alpha[\mathsf{n}_T]([\mathsf{n}]\psi)) \quad (\textit{if } \mathsf{n} \in \mathcal{N}_2 \textit{ and } \alpha \in Act_T)$$

With the unsound translation (viz. Proposition 5.3):

$$\tau_2'([\mathsf{n}]\,\Box_\alpha\,\psi) = \tau_2'(\Box_\alpha[\mathsf{n}]([\mathsf{n}_T]\psi)) \quad (\textit{if } \mathsf{n} \in \mathcal{N}_2 \textit{ and } \alpha \in Act_T)$$

*We have that $\tau_2'(\varphi)$ is **not** well-defined for every input $\varphi \in \mathcal{L}_2$.*

Proof. Let $\mathsf{n} = (\{\alpha\}, \Box_\alpha p, +p, \varnothing)$. We show that τ_2' on input $([\mathsf{n}]\,\Box_\alpha\,p) \in \mathcal{L}_2$ is not well-defined. We have the following:

$$\begin{array}{rcll} \tau_2'([\mathsf{n}]\,\Box_\alpha\,p) & = & \tau_2'(\Box_\alpha[\mathsf{n}]([\mathsf{n}_T]p)) & = \\ \Box_\alpha\tau_2'([\mathsf{n}]\tau_2'([\mathsf{n}_T]p)) & = & \Box_\alpha\tau_2'([\mathsf{n}](p \vee \Box_\alpha p)) & = \\ \Box_\alpha(\tau_2'([\mathsf{n}]p) \vee \tau_2'([\mathsf{n}]\,\Box_\alpha\,p)) & = & \Box_\alpha(p \vee \tau_2'([\mathsf{n}]\,\Box_\alpha\,p)) & \end{array}$$

Thus, we have the following:

$$\begin{gathered} \tau_2'([\mathsf{n}]\,\Box_\alpha\,p) = \cdots = \\ \Box_\alpha(p \vee \tau_2'([\mathsf{n}]\,\Box_\alpha\,p)) = \cdots = \\ \Box_\alpha(p \vee \Box_\alpha(p \vee \tau_2'([\mathsf{n}]\,\Box_\alpha\,p))) = \cdots = \\ \Box_\alpha(p \vee \Box_\alpha(p \vee \Box_\alpha(p \vee \tau_2'([\mathsf{n}]\,\Box_\alpha\,p)))) = \cdots = \\ \Box_\alpha(p \vee \Box_\alpha(p \vee \Box_\alpha(p \vee \Box_\alpha(p \vee \Box_\alpha(p \vee \tau_2'([\mathsf{n}]\,\Box_\alpha\,p)))))) = \ldots \end{gathered}$$

Which continues forever, i.e., function τ_2' on input $[\mathsf{n}]\,\Box_\alpha\, p$ is not well-defined. To briefly show that τ_2 **is** well defined on input $[\mathsf{n}]\,\Box_\alpha\, p$, observe the following:

$$\begin{array}{rcl} \tau_2([\mathsf{n}]\,\Box_\alpha\, p) & = & \tau_2(\Box_\alpha[\mathsf{n}_T]([\mathsf{n}]p)) \quad = \\ \Box_\alpha\tau_2([\mathsf{n}_T]\tau_2([\mathsf{n}]p)) & = & \Box_\alpha\tau_2([\mathsf{n}_T]p) \quad = \\ \Box_\alpha\tau_1([\mathsf{n}_T]p) & = & \Box_\alpha(p \vee \Box_\alpha p) \end{array}$$

□

In the next section we show how we can apply this system to deduce the validity of assertions from this language. Moreover, we again show that this logic is decidable.

5.4.4 Derivations and decidability

In this section we can show how we can use system **L2** in order to derive the validity of assertions from language $\mathcal{L}_2$, and show that this is decidable. Let $\mathsf{n} = (\{1\}, \top, +p, \{2\})$ be the norm which states that, after the performance of action 1, for all states p will hold until action 2 has been performed. We will show the validity of the following assertion:

$$\vdash_{\mathrm{L}_2} [\mathsf{n}](\Box_2\, \Box_1\, p)$$

We again use notation 'RE(x,y)' to denote that we replaced a sub-formula from line x by using an acquired equivalence from line y, which is short-hand notation for applying the rule of replacement of equivalence formulas (RE) and then applying modus ponens (MP) on the result. Moreover, we use the notation 'MP(x,y)' to denote that we apply modus ponens (MP) with the implication from line x and the precedent from line y, and use 'NEC(x)' to denote the application of the necessitation rule on line x:

1.	$[\mathsf{n}]p \leftrightarrow p$	Ax.2.1
2.	$[\mathsf{n}](\Box_1 p) \leftrightarrow \Box_1[(\top, +p, \{1,2\})]([\mathsf{n}]p)$	Ax.2.4
3.	$[\mathsf{n}](\Box_1 p) \leftrightarrow \Box_1[(\top, +p, \{1,2\})]p$	RE(2,1)
4.	$[(\top, +p, \{1,2\})]p \leftrightarrow (\top \vee p)$	Ax.1.1(a)
5.	$[\mathsf{n}](\Box_1 p) \leftrightarrow \Box_1(\top \vee p)$	RE(3,4)
6.	$[\mathsf{n}](\Box_2\, \Box_1\, p) \leftrightarrow \Box_2[\mathsf{n}](\Box_1 p)$	Ax.2.5
7.	$[\mathsf{n}](\Box_2\, \Box_1\, p) \leftrightarrow \Box_2\, \Box_1\, (\top \vee p)$	RE(6,5)
8.	$\top \vee p$	Taut.
9.	$\Box_1(\top \vee p)$	NEC(8)
10.	$\Box_2\, \Box_1\, (\top \vee p)$	NEC(9)
11.	$[\mathsf{n}](\Box_2\, \Box_1\, p)$	MP(7,10)

We can again ask whether $\mathcal{L}_2$ with corresponding system **L2** is *decidable*, i.e. whether there exists an effective procedure that tells us whether an arbitrary formula $\varphi \in \mathcal{L}_2$ is valid. We have the following result.

Theorem 5.7. *The logical language $\mathcal{L}_2$ with corresponding system* **L2** *is decidable.*

Proof. We use the translation function $\tau_2 : \mathcal{L}_2 \to \mathcal{L}_0$ from our completeness proof in Theorem 5.5 to establish the following correspondence:

$$\vdash_{\mathrm{L}_2} \varphi \quad \Leftrightarrow \quad \vdash_{\mathrm{L}_2} \tau_2(\varphi) \quad \Leftrightarrow \quad \vdash_{\mathrm{L}_0} \tau_2(\varphi)$$

Thus, in order to determine whether $\vdash_{\mathrm{L}_2} \varphi$, it is both necessary and sufficient to show that $\vdash_{\mathrm{L}_0} \tau_2(\varphi)$. However, since logic $\mathcal{L}_0$ with corresponding system **L0** is decidable and since τ_2 is well defined and can be effectively computed, decidability again transfers to our logic and system. □

Again, this proof highlights an easier way to establish the validity of a sentence. To show that $\vdash_{\mathrm{L}_2} [\mathsf{n}](\Box_2 \Box_1 p)$, we can perform the following translation:

$$\begin{aligned}
\tau_2([\mathsf{n}](\Box_2 \Box_1 p)) &= \tau_2(\Box_2[\mathsf{n}](\Box_1 p)) = \\
\Box_2 \tau_2([\mathsf{n}](\Box_1 p)) &= \Box_2 \tau_2(\Box_1[\mathsf{n}_T]([\mathsf{n}]p)) = \\
\Box_2 \Box_1 \tau_2([\mathsf{n}_T]([\mathsf{n}]p)) &= \Box_2 \Box_1 \tau_2([\mathsf{n}_T]\tau_2([\mathsf{n}]p)) = \\
\Box_2 \Box_1 \tau_2([\mathsf{n}_T]p) &= \Box_2 \Box_1 (\top \vee p)
\end{aligned}$$

To easily establish that:

$$\vdash_{\mathrm{L}_0} \Box_2 \Box_1 (\top \vee p)$$

This concludes our work on the logical language $\mathcal{L}_2$ and **L2**, extending $\mathcal{L}_1$ and **L1**. We saw that even though our second logic extends the first, decidability is still guaranteed. As we argued in the beginning of this chapter, these logics can be used to reason about dynamic normative systems.

5.5 Discussion

In this chapter we provided a new dynamic modal logic that is able to characterize the dynamics of norm addition of various classes of norms. We have shown that this logic is both sound and complete, which means that we can use automated theorem provers to prove properties of a normative system in a dynamic environment. This framework is of crucial importance in order to perform formal verification of dynamic normative multi-agent system.

This chapter can be extended in multiple ways, which we will briefly reflect upon. First of all, it can be interesting to apply this framework of dynamic update to other well-known frameworks of normative systems, such as the coloured systems considered in [Sergot, 2007]. Secondly, we can look at other classes of norms, such as norms with deadlines, and see how we can reason about these norms in our current model. Lastly, the relation between the current framework of dynamic normative systems and other frameworks of

(dynamic) normative theory needs to be explored further. In the field of deontic logic as briefly introduced in Chapter 2 (i.e. the logic that concerns itself with what is obligatory and prohibited), we often find that viewing a normative system as a static entity may result in several paradoxes, such as is the case with contrary-to-duty obligations. The exact relation between this novel dynamic framework and the evidential paradoxes found in the study of deontic logic needs to be further explored.

Chapter 6

Modelling and Verifying Monitors

In order to control the behaviour of the system and the participating agents, it is not only important to consider the norms, but also the monitors that are able to detect whenever a violation occurs. In other words, both the norms and the monitors serve as a control mechanism for the multi-agent system, and this chapter will focus on the latter. Monitors are important whenever the system as a whole is not fully observable. A natural question that arises is how we can verify whether a monitor is sufficient for a given normative system, that is, whether we can prove that a monitor is sufficient in order to be able to detect the possible norm violations that can occur. This will be the topic of this chapter.

6.1 Introduction

In the research field of normative multi-agent systems, it is often assumed that monitors are perfect in the sense that they can fully observe all agents' activities and correctly evaluate them with respect to a given set of norms. For example, consider the scenario where two cars approach a road bottleneck from opposite directions at the same time. Suppose the norm is that cars coming from the right have priority and that cars cannot pass the road bottleneck at the same time. Existing approaches assume that all cars' activities can be monitored, e.g., for each car it can be observed if the car is just before, within, and just after the road bottleneck. However, this is not a realistic assumption as monitors often can only observe a limited set of events or activities. For example, a monitor may only be able to observe that the car coming from the right is just before the road bottleneck or that a car is within the road bottleneck. In general, the notion of norm monitors is neither defined nor

analysed such that it is not possible to study whether and to which extent a monitor can be effective in detecting some norm violations. The existing (perfect) monitors are inherently coupled to a set of norms in the sense that for a given set of norms the corresponding monitor can observe all agents' activities and evaluate them with respect to the given set of norms.

In this chapter we aim at relaxing these assumptions by providing a logical and computational framework to specify various types of norm monitors. The framework is compositional in the sense that basic monitors can be composed to build more complex and powerful norm monitors. We study formal properties of monitors in order to determine whether a specific monitor can be effective for a specific set of norms. We also consider computational problems of norm monitors such as whether a given monitor can detect violations of a specific set of norms, whether there exists such a monitor, and if there exists a combination of some given monitors to detect the violation of a given set of norms.

This work is related to the work of [Bauer et al., 2011], the crucial difference being that we study *offline* verification, while they study *run-time* verification. The main problem formulated in these works is how to monitor certain temporal logic properties at run-time, and as such, they consider finite runs of possibly increasing size. This work is also related to studies that aim at monitoring constraints (closely related to norms) on databases over the course of time, for example the work found in [Lipeck and Saake, 1987]. The main aim of this work is to check the integrity constraints on a database with minimum knowledge about its history and possible future. Our work can be seen as a more general framework for analysing monitors with respect to a set of constraints that needs to be monitored. This chapter is an extended work of our earlier published work found in [Bulling et al., 2013].

6.1.1 The verification problem

The verification problem we consider in this chapter is the following: Given the specification of a multi-agent system, a specification of the norms and a specification of the monitor(s), verify whether the monitors are sufficient for the norm, that is, verify whether the monitors correctly and completely detect all the possible violations that can occur. The corresponding synthesis and design question is whether we can synthesize such a monitor for a given multi-agent system and norm.

6.1.2 Chapter outline

We give a brief outline of the contents of this chapter:

- In **Section 6.2** (p 155) we introduce the model of execution we are

interested in, define the notion of run-based norms, and define how we can specify monitors.

- In **Section 6.3** (p 160) we introduce the logic of Linear-time Temporal Logic (LTL), and show how we can express run-based norms using formulas from LTL.
- In **Section 6.4** (p 163) we show how we can specify several types of monitors using formulas from LTL. Moreover, we show how we can combine individual monitors into more complicated combined monitors, which are monitors that are able to share their observations.
- In **Section 6.5** (p 171) we discuss and prove several properties of LTL-based monitors, which will later be used to solve several decision problems associated to sound- and completeness of monitors.
- In **Section 6.6** (p 178) we ask several model checking questions and give several computational complexity results.
- In **Section 6.7** (p 181) we discuss this chapter.

6.2 Preliminaries

In this section we introduce the model of execution we are interested in, define the notion of run-based norms and monitors. Moreover, we show several properties of monitors, and show the interplay between systems, norms and monitors.

6.2.1 Transition systems

In this thesis we considered several models of agent execution, depending on what we were interested in. For example, an important consideration would be whether or not we are interested in the agents present in the system, or whether or not we are interested in the possible (joint) actions that the agents can perform. In this chapter, we want to generalize these models, since our focus will be on *monitors* which can be present in any of such systems that we considered in this thesis. The most general kind of systems underpinning all these models of execution are basic *transition systems*. Such systems, which we will formally define shortly, consists of states and transitions between states. Such a transition might corresponds to an action of an agent or a joint action of a group of agents an agent. Since we want to prove and validate certain properties of a system, we add a valuation function as usual, that for each state assigns a set of atomic propositions which are true at this state. We call a

transition system together with a valuation function an *interpreted transition system.*

Definition 6.1 (Interpreted transition system)**.** *An* interpreted transition system *is a tuple* $S = (Q, \Pi, \mu, \rightarrow, q_0)$ *such that:*

- Q *is a finite set of states.*
- Π *is a finite set of atomic propositions.*
- $\mu : Q \rightarrow \mathcal{P}(\Pi)$ *is a valuation function mapping a state* $q \in Q$ *to an element from* $\mathcal{P}(\Pi)$*.*
- $\rightarrow \subseteq Q \times Q$ *is a binary serial relation over* Q*, i.e. for all* $q \in Q$ *there is a state* $q' \in Q$ *such that* $q \rightarrow q'$*.*
- $q_0 \in Q$ *is the initial state.*

In this chapter, we will refer to such a system as simply a transition system, since we only focus our attention on interpreted systems. Note that we use the same symbol S for transition systems and concurrent game structures considered in Chapter 3, although this should lead to no confusion since we only consider transition systems from here-on. Relating to the various properties of multi-agent systems we discussed in Chapter 2, this model is thus synchronous, decentralized, discrete and deterministic. As we already showed in previous chapters, such a transition system gives rise to a set of possible runs.

Definition 6.2 (Runs)**.** *Given a transition system* $S = (Q, \Pi, \mu, \rightarrow, q_0)$ *a* run *of the system is defined as an infinite sequence of states* $q_0 q_1 \ldots \in Q^\omega$ *such that* q_0 *is the starting state and* $\forall n \in \mathbb{N}_0 : (q_n, q_{n+1}) \in \rightarrow$*. For a given run* r*, we define* $r[n]\,(n \geq 0)$ *as usual, i.e. as the* n*-th state* q_n *occurring on the run. Moreover, we define* $r[n, \infty]\,(n \geq 0)$ *as the postfix of the run starting from position* n*. The set of all possible runs over* S *is denoted by* $\mathcal{R}_S$*.*

Again, the focus of this chapter is on *monitors.* This means that we not only want to consider a very general model of execution, but also a very general model of norms, called run-based norms. This will be discussed in the next section.

6.2.2 Run-based norms

In this thesis we considered a multitude of different classes of norms, noteworthy state-based norms, transition-based norms and sequence-based norms. In this chapter however, we want to *generalize* over all these different kinds of

norms. Generally speaking, a (set of) norms gives rise to a set of behaviours of the system which is considered desired and a set of behaviours which considered undesired. Thus, a general way of viewing a norm is just by viewing it as some subset of all the possible behaviours that can occur, where a behaviour of a system is understood as a possible run that can occur. These *run-based* norms were already briefly introduced in Chapter 2, and we formally define them as follows.

Definition 6.3 (Run-based norms)**.** *Given a transition system S, a* run-based norm $\mathcal{R}_{norm}$ *is a subset of system runs, i.e.* $\mathcal{R}_{norm} \subseteq \mathcal{R}_S$*. A run r* complies with a norm $\mathcal{R}_{\text{norm}}$*, for short r* is $\mathcal{R}_{\text{norm}}$-compliant*, if and only if* $r \in \mathcal{R}_{norm}$*. Otherwise, r* violates $\mathcal{R}_{\text{norm}}$*, for short r* is a $\mathcal{R}_{\text{norm}}$-violation*.*

Note that, since the set of runs $\mathcal{R}_S$ for a system S can be infinite, the set $\mathcal{R}_{\text{norm}}$ can also be infinite. Again, we stress that in this chapter we are not interested in the underlying deontic specification of such a norm, we merely use the fact that there exists some mechanism which decides whether a run is considered "desired" or "undesired". The set $\mathcal{R}_{\text{norm}}$ then gives us the set of all the runs which are considered "desired".

Next we will introduce the concept of monitors, where a monitor is in the abstract sense understood as a function over possible behaviours of a system. This concept is explained in the next section.

6.2.3 Monitors

A *monitor* observes the behaviours of the system. In this chapter we adopt the view that a monitor for each possible run r constructs a candidate set of runs that represents all the possibilities that might have actually occurred. In the ideal case, if r takes place the monitor should observe r and nothing else. However, due to several reasons, e.g. cost limitations, noise or environmental restrictions, it is not always possible to perfectly monitor the behaviours of the system. In the general case, given a transition system S we model a monitor as a function $m : \mathcal{R}_S \to \mathcal{P}(\mathcal{R}_S)$. Intuitively, given a run $r \in \mathcal{R}_S$ and a monitor m, the set $m(r) \subseteq \mathcal{R}_S$ contains all the runs that the monitor considers possible as being the actual run r.

Definition 6.4 (Monitors)**.** *Given a transition system S, a* monitor m *over system behaviours $\mathcal{R}_S$ is a function $m : \mathcal{R}_S \to \mathcal{P}(\mathcal{R}_S)$. Moreover, we define $R_m \subseteq \mathcal{R}_S \times \mathcal{R}_S$ as the corresponding relation derived from m. That is:*

$$r R_m r' \quad \Leftrightarrow \quad r' \in m(r)$$

We can associate properties with monitors. Consider for example the extreme case in which for a given transition system S a monitor m is defined

as $m(r) = \mathcal{R}_S$ for all $r \in \mathcal{R}_S$. In this particular case, the monitor cannot distinguish anything; for each run of the system it is possible that any arbitrary run might have actually occurred. The other extreme case which we already mentioned is the case in which it holds that $m(r) = \{r\}$ for all $r \in \mathcal{R}_S$. Here we might say that the monitor can distinguish everything; it always observes the actual run of the system. We introduce the following properties of a monitor.

Definition 6.5 (Properties of a monitor)**.** *Let S be a transition system, and let m be a monitor over system behaviours $\mathcal{R}_S$. We say that m is ...*

- *...*correct *iff for all runs $r \in \mathcal{R}_S$ we have $r \in m(r)$.*
- *...*consistent *iff for all runs $r_1, r_2, r_3 \in \mathcal{R}_S$ we have if ($r_1 \in m(r_3)$ and $r_2 \in m(r_3)$) then $r_1 \in m(r_2)$ and $r_2 \in m(r_1)$.*
- *...*ideal *iff for all runs $r \in \mathcal{R}_S$ we have $m(r) = \{r\}$.*
- *... an* equivalence classifier *iff m is correct and consistent.*
- *...*broken *iff there exists a run $r \in \mathcal{R}_S$ such that $m(r) = \varnothing$.*

The property of correctness states that a monitor never excludes the true run from the set of possibilities. For example, an incorrect monitor may observe something that never actually took place (for example, a noisy camera). The property of consistency states that for any two alternatives in a monitoring observation, the monitor cannot differentiate these runs if any of these alternatives would be the actual run. The underlying idea here is that if a monitor would be able to differentiate them, then this monitor has the capability to always distinguish these runs for every other observation, i.e. these runs "should never appear together". When a monitor is broken, it can be seen as a monitor that makes impossible observations; none of the runs are deemed similar to the run that was observed. We can make several connections between properties of a monitor m and between properties of the underlying relation R_m constituting this monitor.

Proposition 6.1. *Let m be a monitor over system behaviours $\mathcal{R}_S$ for a given transition system S. A monitor m is ...*

- *... broken iff R_m is not serial.*
- *... correct iff R_m is reflexive.*
- *... consistent iff R_m is Euclidian.*
- *... ideal iff R_m is the identity relation.*
- *... an equivalence classifier iff R_m is an equivalence relation.*

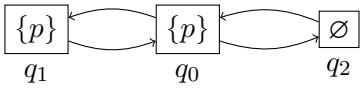


Figure 6.1: A transition system with three states. We consider the norm which tells us that it is forbidden to ever visit state q_2.

This proposition holds since the definitions of broken, correct, consistent, ideal and an equivalence classifier directly coincide with the aforementioned properties of binary relations. However, this allows us to use results found in the study of binary relations to properties of monitors. For example, we can establish the following.

Proposition 6.2. *Given a transition system S and monitor m. We can establish the following relations:*

$$\begin{aligned} m \text{ is ideal} &\Rightarrow m \text{ is an equivalence classifier} \\ m \text{ is an equivalence classifier} &\Rightarrow m \text{ is correct} \\ m \text{ is an equivalence classifier} &\Rightarrow m \text{ is consistent} \\ m \text{ is correct} &\Rightarrow m \text{ is not broken} \end{aligned}$$

Certainly more relations can be shown, but that is beyond the scope of this chapter. We end this section with an example, which shows an example transition system with an example monitor and an example norm. This example highlights the interplay that can occur between these components.

Example 6.1. *Consider the transition system S shown in Figure 6.1, with states $Q = \{q_0, q_1, q_2\}$, where q_0 is the initial state of the system. Let $\mathcal{R}_{norm} = \{(q_0\, q_1)^\omega\}$ consist of a single run which only visits q_0 and q_1 in succession. This norm states that it is forbidden to ever visit state q_2. We define the following monitor:*

$$m(r) = \{r' \in \mathcal{R}_S \mid r[1] = r'[1]\}$$

This monitor observes only the first step of a possible execution, and partitions the runs into two distinct classes: one class consists of all the runs that initially visit state q_1 and the other one of all the runs that initially visit q_2. This monitor is not broken, correct and consistent, but it is clearly not ideal. If r initially visits q_2, then all the runs in the set $m(r)$ are $\mathcal{R}_{norm}$-violations. However, if r initially visits q_1, exactly 1 run in the set $m(r)$ is not an $\mathcal{R}_{norm}$-violation while the remaining runs are.

In this section we defined norms and monitors as very abstract entities. We defined norms as *some* subset of runs, and monitors as *some* function over these runs. In the next section we will introduce Linear-time Temporal Logic as a more concrete (but still very general) way to characterize these norms and monitors.

6.3 Linear-time Temporal Logic

In the following we introduce *Linear-time Temporal Logic* (LTL) as introduced in [Pnueli, 1977]. Formulas of LTL are defined by the following grammar, where $p \in \Pi$ denotes an atomic proposition:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \bigcirc\varphi \mid \varphi \mathcal{U} \varphi$$

LTL-formulas are interpreted over a run r of a transition system S. Intuitively, $\bigcirc\varphi$ ("next φ") means that φ holds in the next state of the run, and $\varphi_1 \mathcal{U} \varphi_2$ that φ_1 holds along the run until φ_2 holds ("φ_1 until φ_2"). We define $\Diamond\varphi \equiv \top\mathcal{U}\varphi$ ("eventually φ") and $\Box\varphi \equiv \neg \Diamond \neg\varphi$ ("always φ") as short-hand formulas in the usual way. Formally, a formula of this language is evaluated along a transition system and a run (or along *a set of runs*) in the following way.

Definition 6.6 (LTL semantics)**.** *Given a transition system $S = (Q, \Pi, \mu, \rightarrow, q_0)$, a run $r \in \mathcal{R}_S$ and a set of runs $\mathcal{R}' \subseteq \mathcal{R}_S$, the satisfaction relation $\models_{LTL}$ of LTL is inductively defined as follows:*

- $S, r \models_{LTL} p$ *iff* $p \in \mu(r[0])$
- $S, r \models_{LTL} \neg\varphi$ *iff* $S, r \not\models_{LTL} \varphi$
- $S, r \models_{LTL} \varphi_1 \vee \varphi_2$ *iff* $S, r \models_{LTL} \varphi_1$ *or* $S, r \models_{LTL} \varphi_2$
- $S, r \models_{LTL} \bigcirc\varphi$ *iff* $S, r[1, \infty] \models_{LTL} \varphi$
- $S, r \models_{LTL} \varphi_1 \mathcal{U} \varphi_2$ *iff there exists an* $i \geq 0$ *such that* $S, r[i, \infty] \models_{LTL} \varphi_2$*, and for all* $0 \leq k < i$ *we have that* $S, r[k, \infty] \models_{LTL} \varphi_1$
- $S, \mathcal{R}' \models_{LTL} \varphi$ *iff* $\forall r \in \mathcal{R}' : S, r \models_{LTL} \varphi$

In the remainder of this chapter we write $\models$ for $\models_{\text{LTL}}$ whenever clear from context. Note that we consider lifted semantics for LTL, i.e. we can either evaluate a formula along a run or along a set of runs. The reason for the latter will become apparent in the next section where we will use this logic to reason about the capabilities of a monitor.

6.3.1 LTL-based norms

A run-based norm $\mathcal{R}_{\text{norm}}$ can be defined by an LTL-formula ψ such that $\mathcal{R}_{\text{norm}}$ contains all the runs and only the runs satisfying ψ, allowing for a very concise representation of such a norm.

Definition 6.7 (LTL-based norms)**.** *Given a transition system S and an LTL-formula ψ, we let $\mathcal{R}_S(\psi) = \{r \in \mathcal{R}_S \mid S, r \models \psi\}$ be defined as all the runs in S satisfying ψ. Whenever we refer to a formula ψ as a norm for a given transition system S, we imply that $\mathcal{R}_{norm} = \mathcal{R}_S(\psi)$.*

Note however that in the general case, not all run-based norms can be described by an LTL formula. Let S be a model, ψ be a norm, and m be a monitor. Now we are ready to reason about norm violations. As said in the previous section, a run r is a $\mathcal{R}_{\text{norm}}$-violation if and only if $r \notin \mathcal{R}_{\text{norm}}$, that is, if and only if $r \notin \mathcal{R}_S(\psi)$. In this setting, this is equivalent to saying that r is a $\mathcal{R}_{\text{norm}}$-violation if and only if $S, r \vDash \neg\psi$. However, this violation may not be detected by monitor m; this heavily depends on the capabilities of the monitor and what it can observe. Intuitively, a norm violation for a run r is detected if in all the possibilities that the monitor considers, a violation occurs, i.e. whenever $m(r) \subseteq \mathcal{R}_S(\neg\psi)$ holds. This statement is equivalent to saying that a violation is detected if $S, m(r) \vDash \neg\psi$. In the case of a non-ideal monitor the classification is not that simple: some runs in $m(r)$ may violate the norm whereas others do not. Formally we can define it as follows.

Definition 6.8 (Detection of a norm violation)**.** *Given a transition system S, a norm ψ, and a monitor m. We say that monitor m on input $r \in \mathcal{R}_S$ detects a ...*

- *...* ψ-violation *iff* $S, m(r) \vDash \neg\psi$*;*
- *...* ψ-compliance *iff* $S, m(r) \vDash \psi$*; and*
- *...* ψ-indifference *iff both* $S, m(r) \nvDash \psi$ *and* $S, m(r) \nvDash \neg\psi$*.*

Thus, a run is evaluated with respect to a given monitor and norm. However, if the monitor detects a possible norm violation, the norm may actually not be violated. Another possibility might be that a monitor does not detect a norm violation, while in fact such a violation did occur. These are considered *classification errors*, which fall into the following categories.

Definition 6.9 (Classification errors)**.** *Given a transition system S, a norm ψ, and a monitor m. We say that m makes a* ψ-classification error *on $r \in \mathcal{R}_S$ iff ...*

- *...m detects a ψ-violation on r and r is not a ψ-violation. Such a classification error is referred to as a false negative; or ...*
- *...m detects a ψ-compliance on r and r is a ψ-violation. Such a classification error is referred to as a false positive.*

Ultimately, we are interested in monitors which detect all norm violations and only those.

Definition 6.10 (Sound monitors, complete monitors, sufficient monitors)**.** *Given a transition system S, a norm ψ and a monitor m. We say that m is ...*

- ... ψ-sound *in* S *iff for all* $r \in \mathcal{R}_S$ *it holds:*

$$S, m(r) \vDash \neg\psi \quad \Rightarrow \quad S, r \vDash \neg\psi$$

- ... ψ-complete *in* S *iff for all* $r \in \mathcal{R}_S$ *it holds:*

$$S, r \vDash \neg\psi \quad \Rightarrow \quad S, m(r) \vDash \neg\psi$$

- ... ψ-sufficient *in* S *iff* m *is* ψ*-sound and* ψ*-complete. That is, iff for all* $r \in \mathcal{R}_S$ *it holds:*

$$S, m(r) \vDash \neg\psi \quad \Leftrightarrow \quad S, r \vDash \neg\psi$$

Again, we omit S *if the model is clear from context.*

In words, a monitor is sound with respect to a norm ψ when for every possible run of the system, it holds that whenever a ψ-violation is detected it is the case that the run was a ψ-violation. A monitor is complete with respect to a norm ψ when for every possible run of the system, it holds that whenever the run was a ψ-violation it is the case that a ψ-violation is detected. A monitor is sufficient with respect to a norm ψ if all violations are correctly detected.

To summarize these concepts, we return to our example transition system shown in Figure 6.1.

Example 6.2. *We return to the transition system found in Example 6.1. The norm can now be specified as an LTL formula* $\psi = \Box p$*, for which we indeed have that* $\mathcal{R}_S(\psi) = \{(q_0 q_1)^\omega\}$*. Let us also consider that we still have the same monitor* m *as defined as follows:*

$$m(r) = \{r' \in \mathcal{R}_S \mid r[1] = r'[1]\}$$

It is now easy to verify that this monitor is ψ*-sound; whenever for a given run* r *it holds that* $S, m(r) \vDash \neg \Box p$ *then certainly* $S, r \vDash \neg \Box p$*. However, this monitor is not* ψ*-complete. In particular, consider the run* $r = q_0 q_1 (q_0 q_2)^\omega$*. It is clear that for this run it holds that* $S, r \vDash \neg \Box p$*, however it also holds that* $S, m(r) \nvDash \neg \Box p$*. This is because the run* $(q_0 q_1)^\omega$ *is also in the set* $m(r)$ *and this run is not a* ψ*-violation. Thus, for this run the monitor detects a* ψ*-indifference, but it should have detected a* ψ*-violation for it to not break the property of* ψ*-completeness.*

In this section we have shown how we can reason about the capabilities of monitors using Linear-time Temporal Logic. In upcoming section we show how we can *define* monitors using this logic.

6.4 LTL-based monitors

In the previous section, we introduced the concept of a monitor as a general function over system runs. In this section we will show that we can also *characterize* monitors using formulas from Linear-time Temporal Logic. The underlying idea is that instead of explicitly defining for each possible run what a monitor can observe, we use a temporal formula to represent what the monitoring capabilities are. Such a formula thus gives rise to a monitoring function, which allows us to apply our framework to these LTL-based monitors as well. We will first introduce basic LTL-monitors. Later we will consider *combined* LTL-monitors, which are monitors built up from multiple basic LTL-monitors. We will end this section with an elaborate example in which we show how we can apply our framework.

6.4.1 Basic LTL-monitors

First of all, we will discuss how we can use LTL formulas in order to characterize monitors. Suppose we have an LTL-formula φ, the resulting φ-*monitor* is then a monitor that (a) can distinguish runs which satisfy φ from those which do not satisfy φ, and (b) cannot distinguish more than that. This is a much simpler characterization of a monitor since it only divides the set of runs into two monitoring classes, particularly the set of runs that satisfy φ and the set of runs that do not satisfy φ. Thus, (a) implies that if for two runs r and r' it holds that if one satisfies φ while the other run does not, then we have that $r' \notin m(r)$ and $r \notin m(r')$, i.e. this monitor can distinguish runs which satisfy φ from runs that do not. Moreover, (b) implies that if both r and r' satisfy φ (or both do not), then it holds that $r' \in m(r)$ and $r \in m(r')$, i.e. this monitor cannot distinguish more than φ. We can formally define such a monitor as follows:

Definition 6.11 (φ-monitor)**.** *Let φ be an LTL-formula and S be a transition system. The* φ-monitor *in S is the function $m_\varphi : \mathcal{R}_S \to \mathcal{P}(\mathcal{R}_S)$ defined as follows:*

$$m_\varphi(r) := \{r' \in \mathcal{R}_S \mid S, r \vDash \varphi \text{ iff } S, r' \vDash \varphi\}$$

We omit transition system S if clear from context.

We briefly return to our previous example to see how we can now associate an LTL-formula with a monitor.

Example 6.3. *We return to the transition system and monitor m found in Example 6.1, which was defined as follows:*

$$m(r) = \{r' \in \mathcal{R}_S \mid r[1] = r'[1]\}$$

From our definition of LTL-monitors, it is now clear that:

$$m = m_{\bigcirc p} = m_{\bigcirc \neg p}$$

That is, we could have defined m as a $\bigcirc p$-monitor or as a $\bigcirc \neg p$-monitor, all of which would return exactly the same monitoring sets for any possible run. That is:

$$\forall r \in \mathcal{R}_S : m(r) = m_{\bigcirc p}(r) = m_{\bigcirc \neg p}(r)$$

This example immediately highlights an important property of an LTL-based monitor, which is that such a specification in general is *not unique*. Particularly, it can be the case that several different LTL-formulas all give rise to the same monitor. A useful insight is that given an LTL-formula φ, the monitor m_φ is always exactly the same monitor as $m_{\neg\varphi}$, as shown in the next proposition

Proposition 6.3. *Given a transition system S and LTL-formula φ. We have the following:*

$$m_\varphi = m_{\neg\varphi}$$

Proof. We have to show that $\forall r \in \mathcal{R}_S$ it holds that $m_\varphi(r) = m_{\neg\varphi}(r)$. Assume an arbitrary $r, r' \in \mathcal{R}_S$, we show that $r' \in m_\varphi(r)$ if and only if $r' \in m_{\neg\varphi}(r)$:

$$r' \in m_\varphi(r) \quad \Leftrightarrow \quad S, \{r, r'\} \vDash \varphi \text{ or } S, \{r, r'\} \vDash \neg\varphi \quad \Leftrightarrow \quad r' \in m_{\neg\varphi}(r)$$

□

Before we show how we can combine several basic LTL-monitors to create more expressive combined monitors, it is useful to consider several *types* of LTL-monitors that we can express using this formalism.

6.4.2 Types of LTL-monitors

In this subsection we will discuss several types of monitors that we can express with LTL-monitors. Below we consider *state-safety* monitors (monitoring that no bad states will ever occur), *transition-safety* monitors (monitoring that no bad transitions will ever occur) and *state-liveness* monitors (monitoring that good states will eventually occur). This list is by no means exhaustive, and merely serves the purpose of getting some more intuition on the various types of monitors that are expressible.

- A **state-safety monitor** is a monitor that, given an arbitrary propositional formula $\varphi \in \mathcal{L}_{\text{prop}}(\Pi)$ describing a state-condition, monitors whether φ is an invariant of the execution, i.e. monitors whether all states that are visited satisfy φ. Such a monitor is given by the function

$m_{\square\varphi}$, and is thus used to monitor safety of an execution over states, i.e. no bad states will occur. An example of such a monitor would be a camera on the motorway monitoring whether all cars are not exceeding the speed limit.

- A **transition-safety monitor** is a monitor that, given two propositional formulas $\varphi_1, \varphi_2 \in \mathcal{L}_{\text{prop}}(\Pi)$ describing two state-conditions, monitors whenever a φ_1-state is visited whether the next occurring state is a φ_2-state. Such a monitor is given by the function $m_{\square(\varphi_1 \to \bigcirc\varphi_2)}$, and is thus used to monitor safety of an execution over transitions, i.e. no bad transitions will occur. Such a monitor can be used to monitor violations of transition-based norms, as discussed in Chapter 3.

- A **state-liveness monitor** is a monitor that, given two propositional formulas $\varphi_1, \varphi_2 \in \mathcal{L}_{\text{prop}}(\Pi)$ describing two state-conditions, monitors whenever a φ_1-state is visited whether a φ_2-state eventually occurs in the future. Such a monitor is given by the function $m_{\square(\varphi_1 \to \Diamond\varphi_2)}$, and is thus used to monitor liveness of an execution over states. Such a monitor can for example check whenever a process is started (φ_1) that this process will eventually end ($\Diamond\varphi_2$), e.g. a sensor monitoring whenever there is a traffic jam that it will eventually disappear or a sensor monitoring whenever a bridge is open that it will eventually close.

Certainly more kinds of LTL-based monitors can be considered. However, we will focus our attention on how various monitors can be *combined* to create monitors that are even more expressive.

6.4.3 Combinations of LTL-monitors

The idea of combining monitors is the underlying idea that monitors can, in some way, combine their individual observations. Suppose we have two monitors m_1 and m_2, who both make for a given transition system S an observation on input $r \in \mathcal{R}_S$, i.e. they both return a set $m_1(r) \subseteq \mathcal{R}_S$ and $m_2(r) \subseteq \mathcal{R}_S$. These sets represent for both monitors the set of alternatives which are deemed possible as being the true run of the system given that r would be the actual run. Now if both these monitors would be able to share their observations, they would be able to eliminate all runs which are in $m_1(r)$, but not in $m_2(r)$, and all runs which are in $m_2(r)$, but not in $m_1(r)$. Thus, the result of sharing this monitoring information would be the set $m_1(r) \cap m_2(r) \subseteq \mathcal{R}_S$. Formally, *combined (LTL-based) monitors* are defined as follows.

Definition 6.12 (Combined LTL-based monitors)**.** *Let S be a transition system and let $m_1, m_2 : \mathcal{R}_S \to \mathcal{P}(\mathcal{R}_S)$ be two monitors. We define the monitor*

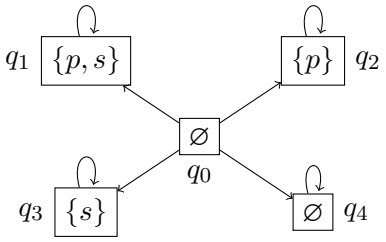


Figure 6.2: A transition system with four possible runs, which allows us to demonstrate that combined monitors can be more expressive than non-combined monitors.

$m_1 \oplus m_2 : \mathcal{R}_S \to \mathcal{P}(\mathcal{R}_S)$ *as follows:*

$$m_1 \oplus m_2(r) := m_1(r) \cap m_2(r)$$

Let Φ *be a non-empty finite set of LTL-formulas. The* combined Φ-monitor *is the function* $m_\Phi : \mathcal{R}_S \to \mathcal{P}(\mathcal{R}_S)$ *such that:*

$$m_\Phi(r) := \bigoplus_{\varphi \in \Phi} m_\varphi(r)$$

From this definition it should be clear that combined LTL-based monitors *generalize* non-combined LTL-based monitors. In particular, we have that $m_\varphi = m_{\{\varphi\}}$ for a given LTL-formula φ. Note that in this work, we do not consider how the information is shared between the monitors, which can for example occur through some communication channel like a distributed network. The following result is clear from the definition of m_Φ.

Proposition 6.4. *The operator* $\oplus$ *is associative and commutative. That is, given arbitrary monitors* m_1*,* m_2 *and* m_3*, we have:*

$$\begin{aligned} \textit{Associative:} \quad & (m_1 \oplus m_2) \oplus m_3 = m_1 \oplus (m_2 \oplus m_3) \\ \textit{Commutative:} \quad & m_1 \oplus m_2 = m_2 \oplus m_1 \end{aligned}$$

Interesting scenarios can for example arise when, given a run r and two monitors m_1, m_2, it is the case that $m_1(r) \neq \varnothing$, $m_2(r) \neq \varnothing$, but for which $m_1(r) \cap m_2(r) = \varnothing$. This would mean that *at least* one of the monitors is *not correct*, since it must hold that $r \notin m_1(r)$ or $r \notin m_2(r)$. However, since we are currently only considering combined LTL-monitors, this scenario can never happen. The reason for this is that (combined or non-combined) LTL-monitors are always correct (viz. Proposition 6.6). To illustrate the concept of combined monitors, we consider the following example.

Example 6.4. *Consider the transition system S shown in Figure 6.2 with initial state q_0. The set of runs is given by $\mathcal{R}_S = \{q_0(q_1)^\omega, q_0(q_2)^\omega, q_0(q_3)^\omega, q_0(q_4)^\omega\}$, and we refer to run $q_0(q_i)^\omega$ with r_i, where $i \in \{1, \ldots, 4\}$. Suppose the norm is given by $\psi = \bigcirc((p \wedge \neg s) \vee (\neg p \wedge s))$, for which we have that $\mathcal{R}_S(\psi) = \{r_2, r_3\}$. Consider the two LTL-monitors $m_{\bigcirc p}$ and $m_{\bigcirc s}$, i.e. the monitor $m_{\bigcirc p}$ checks whether in the next state p holds, and the monitor $m_{\bigcirc s}$ whether in the next state s holds. We have the following:*

- $m_{\bigcirc p}(r_1) = m_{\bigcirc p}(r_2) = \{r_1, r_2\}$
  $m_{\bigcirc p}(r_3) = m_{\bigcirc p}(r_4) = \{r_3, r_4\}$
- $m_{\bigcirc s}(r_1) = m_{\bigcirc s}(r_3) = \{r_1, r_3\}$
  $m_{\bigcirc s}(r_2) = m_{\bigcirc s}(r_4) = \{r_2, r_4\}$

Clearly, none of these monitors is ψ-sufficient. Let $\Phi = \{\bigcirc p, \bigcirc s\}$, then for the monitor m_Φ we have $m_\Phi(r_i) = \{r_i\}$ for $i = 1, \ldots 4$. That is, the monitor perfectly classifies all runs and is ψ-sufficient. Moreover, we have that m_Φ is ideal. On the other hand, it is important to see that the LTL-monitor $m_{\bigcirc p \wedge \bigcirc s}$ is not *ψ-sufficient. In particular, we have:*

- $m_{\bigcirc p \wedge \bigcirc s}(r_1) = \{r_1\}$
  $m_{\bigcirc p \wedge \bigcirc s}(r_2) = m_{\bigcirc p \wedge \bigcirc s}(r_3) = m_{\bigcirc p \wedge \bigcirc s}(r_4) = \{r_2, r_3, r_4\}$

This example highlights an important property that combined monitors can indeed be more expressive than non-combined monitors. The next proposition shows that there are combined monitors for which there is no equivalent non-combined LTL-monitor; that is, combined monitors are strictly more expressive.

Proposition 6.5. *The class of non-combined LTL-monitors is not closed under operator $\oplus$.*

Proof. In Example 6.4 we have constructed a monitor which partitions the set of runs into four distinct equivalence classes, thus if we can show that we cannot construct a non-combined LTL-monitor which can also do this we can conclude our proposition. The actual proof follows from Proposition 6.6 (page 172) later in this chapter, which shows that a non-combined LTL-monitor can partition the set of runs in at most two equivalence classes. □

Although this intuitively already follows from Example 6.4, the actual proof of this proposition is given in Proposition 6.6 where we show that the number of classes a monitor partitions the runs in can be strictly higher for combined monitors than for non-combined monitors. Although combining monitors *can* give you more expressive monitors, it is not *guaranteed* that

this always happens. That is, it might be the case that combining monitors together does *not* give you extra monitoring capabilities. We can show this again with an example.

Example 6.5. *Returning to the transition system S shown in Figure 6.2 and the (ideal) combined monitor $m_{\{\bigcirc p,\bigcirc s\}}$ which we discussed earlier, it is not hard to see that the combination of this monitor with any other monitor will have the same monitoring capabilities. Particularly, for every possible LTL-formula φ, we have that:*

$$m_{\{\bigcirc p,\bigcirc s\}} \oplus m_{\varphi} = m_{\{\bigcirc p,\bigcirc s\}}$$

Although this example shows that combining an ideal monitor with any other monitor will certainly not give you a monitor with more monitoring capabilities, in the general case it is not always clear whether combining monitors together will give you a monitor with more capabilities. This leads to the following definition of *redundant* monitors.

Definition 6.13 (Redundant monitors)**.** *Given a transition system S, an LTL-monitor m_Φ, where Φ is a finite non-empty set of LTL formulas, and an LTL-monitor m_φ, we say that m_φ is* redundant *for m_Φ in S if and only if $m_\Phi \oplus m_\varphi = m_\Phi$.*

Intuitively, we call a monitor m_φ redundant for m_Φ when m_φ does not add any monitoring capabilities to m_Φ. The combination of m_Φ and m_φ will thus give you the exact same monitor m_Φ. This definition leads to the question of how we can determine whether a monitor is redundant. We will show later that, although the problem of determining whether a monitor is redundant and the problem of determining whether a monitor is sufficient seem completely unrelated, these problems are actually very similar. In fact, we will show later that these two seemingly unrelated problems can be expressed in terms of each other.

Before we move to our results regarding properties LTL-based monitors, we will show an elaborate example where we will apply our framework.

6.4.4 Example scenario

In this example we consider that there are two cars approaching a bottleneck in the road, both coming from opposite directions. Moreover, both cars want to reach the opposite side. In order to avoid the dangerous situation where both cars are on the bottleneck simultaneously, there is a norm in place that states the following:

1. When two cars are on opposite sides of the bottleneck, cars coming from the right have priority over cars coming from the left, and;

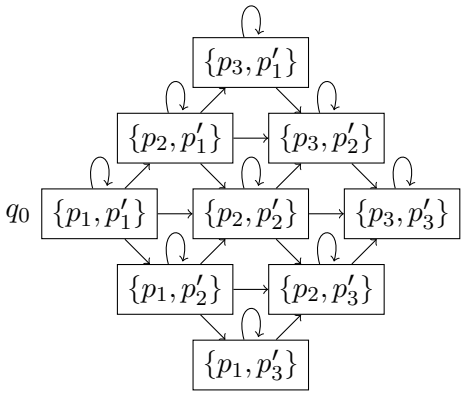


Figure 6.3: Transition system describing the situation where there are two cars on opposite sides of a bottleneck, and both cars from initial situation $q_0 = \{p_1, p_1'\}$ want to reach the opposite side $\{p_3, p_3'\}$.

2. No two cars should simultaneously be on the bottleneck.

We can associate with each car three different general locations. Particularly, (1) just before the bottleneck; (2) on the bottleneck; and (3) on the other side of the bottleneck. We denote each of these situations with a propositions p_1, p_2 and p_3 (p_1 for situation 1, p_2 for situation 2, and p_3 for situation 3) for the car coming from the left, and (1) p_1', (2) p_2' and (3) p_3' for the car coming from the right. The state where both cars are just before the bottleneck is thus given by $\{p_1, p_1'\}$, and the state where both cars reach the opposite side by $\{p_3, p_3'\}$. Assuming that the cars can either wait or move forward (and not turn in the opposite direction), the transition system is shown in Figure 6.3, where we let the initial state q_0 be equal to $\{p_1, p_1'\}$.

We now move to the temporal logic based setting. The LTL-norm corresponding to the situation described above is the following:

$$\psi \equiv \Box((p_1 \wedge p_1') \rightarrow \neg\bigcirc p_2) \wedge \Box\neg(p_2 \wedge p_2').$$

In words, when both cars on opposite sides of the bottleneck (p_1 and p_1' are true), then cars coming from the right have priority over cars coming from the left (it should not be the case that in the next state, the car coming from the left is on the bottleneck, i.e. $\neg\bigcirc p_2$). Moreover, it should never be the case that both cars are on the bottleneck simultaneously (p_2 and p_2' are true). We now consider several example monitors:

Example 6.6. *Consider the state-safety monitor $m_{\Box(p_3 \rightarrow p_3')}$. This monitor is intuitively understood as a monitor that checks whether the car from the right*

crosses the bottleneck before the car from the left does. For each run going through the state (p_3, p_1') *or* (p_3, p_2') *it holds that this monitor detects a* ψ*-violation. To see this, each of the runs that satisfy* $\neg\,\square\,(p_3 \rightarrow p_3')$ *go from state* $\{p_1, p_1'\}$ *to state* $\{p_2, p_1'\}$*, or go through the state* $\{p_2, p_2'\}$*. In other words, this monitor detects* some ψ*-violations that can occur. However, this monitor is not* ψ*-complete, i.e. it does not detect* all ψ*-violations. In particular, the run* $\{p_1, p_1'\}\{p_2, p_2'\}(\{p_3, p_3'\})^\omega$ *is not detected, even though it violates the norm.*

This example highlights that monitors which may initially seem to be unrelated to the norm, can still be used to detect some violations. This again highlights the need for a rigorous framework in order to verify what the capabilities of a monitor is with respect to a norm. In the next example we show how we can combine monitors which are strictly more expressive.

Example 6.7. *Consider the following two monitors* m_1 *and* m_2*, where* m_1 *is a transition-safety monitor and* m_2 *a state-safety monitor:*

$$m_1 = m_{\square(p_1 \rightarrow \bigcirc(\neg p_2 \vee \neg p_2'))}$$
$$m_2 = m_{\square\neg(p_2 \wedge p_1')}$$

Monitor m_1 *is intuitively understood as the monitor that checks if we are in a situation where the car coming from the left is still at the start of the bottleneck (*p_1 *is true), whether in the next state the cars are not simultaneously on the bottleneck (either* p_2 *or* p_2' *is false). Monitor* m_2 *is intuitively understood as the monitor that checks whether it is never the case that the car coming from the left is on the bottleneck (*p_2*) while the car coming from the right is still at the start (*p_1'*). The monitor* m_1 *detects* some ψ*-violations, for example on input* $\{p_1, p_1'\}\{p_2, p_2'\}(\{p_3, p_3'\})^\omega$ *the monitor detects a* ψ*-violation. However, it does not detect* all ψ*-violations, for example the run* $\{p_1, p_1'\}(\{p_2, p_1'\})^\omega$ *which is a* ψ*-violation. The monitor* m_2 *also detects* some ψ*-violations, particularly it detects a* ψ*-violation on input* $\{p_1, p_1'\}(\{p_2, p_1'\})^\omega$*, but it does not detect* all ψ*-violations, namely the run* $\{p_1, p_1'\}\{p_2, p_2'\}(\{p_3, p_3'\})^\omega$ *is not detected. However, by combining the monitors* m_1 *and* m_2 *into* $m_1 \oplus m_2$*, we achieve a monitor which is* ψ*-sufficient. That is, this monitor detects* all ψ*-violation that can occur.*

This latter example again highlights that by combining monitors, we can construct monitors with strictly higher monitoring capabilities. Particularly, we were able to combine monitors which were individually not ψ-sufficient into a monitor which was.

In this section we provided an example scenario in which we highlighted the need for a rigorous framework in order to verify what the capabilities of a monitor is with respect to a norm. Moreover, we showed how we can combine

monitors which are strictly more expressive. The next section will be about the logical properties of these (combined and non-combined) LTL-monitors. This section will be important, since it will introduce several theorems which will then later be used to solve several decision problems associated to sound- and completeness of monitors.

6.5 Monitoring and properties

In this section we discuss and prove several properties of LTL-based monitors, which we will use later to solve several decision problems associated to sound- and completeness of monitors. We will mainly focus on result for combined monitors, since they generalize non-combined monitors. Of utmost importance is the insight that LTL-monitors *partition* the set of runs in a system. We therefore introduce the following notation.

Definition 6.14 (Equivalence classes)**.** *Given a transition system S and a monitor m, whenever m is an equivalence classifier (that is, whenever m is correct and consistent), we denote with $\mathcal{C}(m)$ the set of equivalence classes of m over $\mathcal{R}_S$:*

$$\mathcal{C}(m) := \{m(r) \mid r \in \mathcal{R}_S\}$$

Given a finite set of LTL-formulas Φ, it will be useful to reason about the set of all possible conjunctions we can make by taking for each formula $\varphi \in \Phi$ either φ itself or $\neg\varphi$. We use $\mathsf{Con}(\Phi)$ to denote the set of all conjunctions of formulas from Φ we can construct this way, and define it formally as follows.

Definition 6.15 (Set of conjunctions)**.** *Given an arbitrary finite set of LTL formulas Φ. Whenever $\Phi = \varnothing$, we let $\mathsf{Con}(\Phi) = \{\bot\}$. Whenever $\Phi = \{\varphi_1, \ldots, \varphi_n\}$, we define $\mathsf{Con}(\Phi)$ as:*

$$\mathsf{Con}(\Phi) := \{\varphi'_1 \wedge \cdots \wedge \varphi'_n \mid \varphi'_i \in \{\varphi_i, \neg\varphi_i\} \text{ for } i = 1, \ldots, n\}$$

Thus, each conjunction $\hat{\varphi} \in \mathsf{Con}(\Phi)$, contains for each formulas $\varphi \in \Phi$ either φ itself or $\neg\varphi$. Since this notation will be of importance later, we also give an example.

Example 6.8. *Suppose we have the set $\Phi = \{\varphi_1, \varphi_2\}$. We have:*

$$\mathsf{Con}(\Phi) = \{\ \varphi_1 \wedge \varphi_2\ ,\ \varphi_1 \wedge \neg\varphi_2\ ,\ \neg\varphi_1 \wedge \varphi_2\ ,\ \neg\varphi_1 \wedge \neg\varphi_2\ \}$$

From this example, it is clear that for a given (finite) set Φ, the set $\mathsf{Con}(\Phi)$ contains exactly $2^{|\Phi|}$ elements. The following proposition highlights why the set of conjunctions play such an important role in combined monitors.

Proposition 6.6. *For each system S and finite set of LTL-formulas Φ, the monitor m_Φ is an equivalence classifier with $1 \leq |\mathcal{C}(m_\Phi)| \leq 2^{|\Phi|}$.*

Proof. From our definition of $\mathsf{Con}(\Phi)$, it is clear that we have the following:

$$\bigcup_{\hat{\varphi} \in \mathsf{Con}(\Phi)} \mathcal{R}_S(\hat{\varphi}) = \mathcal{R}_S$$

Given $\hat{\varphi}, \hat{\varphi}' \in \mathsf{Con}(\Phi)$, if $\hat{\varphi} \neq \hat{\varphi}'$, then $\hat{\varphi} \wedge \hat{\varphi}'$ is not satisfiable in S, since we can always find a $\varphi \in \Phi$ which is true in $\hat{\varphi}$ and false in $\hat{\varphi}'$, or vice versa. This implies that $\mathcal{R}_S(\hat{\varphi}) \cap \mathcal{R}_S(\hat{\varphi}') = \varnothing$, which allows us to conclude that $\{\mathcal{R}_S(\hat{\varphi}) \subseteq \mathcal{R}_S \mid \hat{\varphi} \in \mathsf{Con}(\Phi)\} \backslash \{\varnothing\}$ forms a partition over $\mathcal{R}_S$. Now assume an arbitrary satisfiable $\hat{\varphi}' = \varphi'_1 \wedge \cdots \wedge \varphi'_n \in \mathsf{Con}(\Phi)$ and run $r \in \mathcal{R}_S(\hat{\varphi}')$. Then, for all runs $r' \in \mathcal{R}_S$ we have: $r' \in m_\Phi(r)$ if and only if $S, r' \vDash \varphi'_1$ and $\ldots$ and $S, r' \vDash \varphi'_n$ if and only if $S, r' \vDash \hat{\varphi}'$ if and only if $r' \in \mathcal{R}_S(\hat{\varphi}')$. This shows that $m_\Phi(r) = \mathcal{R}_S(\hat{\varphi}')$, implying that m is an equivalence classifier. Moreover, for every $\hat{\varphi} \in \mathsf{Con}(\Phi)$ we have that if this formula is satisfiable in S, then this formula represents an equivalence class of the monitor. That is:

$$\mathcal{C}(m) = \{\mathcal{R}_S(\hat{\varphi}) \subseteq \mathcal{R}_S \mid \hat{\varphi} \in \mathsf{Con}(\Phi)\} \backslash \{\varnothing\}$$

The number of satisfiable formulas in $\mathsf{Con}(\Phi)$ is at least 1 and at most $2^{|\Phi|}$, thus we have $1 \leq |\mathcal{C}(m_\Phi)| \leq 2^{|\Phi|}$. □

As already mentioned in Proposition 6.5, this proposition can be used to show that the class of non-combined monitors is not closed under operator $\oplus$. For a non-combined monitor m_φ, we thus have that $1 \leq |\mathcal{C}(m_\varphi)| \leq 2$. An example of a monitor that only partitions the runs into one single equivalence class independent of the transition system would be $m_\top$ or $m_\bot$, i.e. $|\mathcal{C}(m_\top)| = |\mathcal{C}(m_\bot)| = 1$. Using the result that each LTL-based monitor is an equivalence classifier, the next proposition allows us to elegantly show that if an LTL-based monitor is ψ-sufficient in a given transition system, then this monitor is also $\neg\psi$-sufficient. That is to say, the set of sufficient monitors for a given norm is closed under complement of the norm.

Proposition 6.7. *Given a transition system S, an LTL-norm ψ and monitor m_Φ, where Φ is a non-empty finite set of LTL formulas. We have:*

$$m_\Phi \text{ is } \psi\text{-sufficient in } S \iff m_\Phi \text{ is } \neg\psi\text{-sufficient in } S$$

Proof. We only have to show one direction, since the other direction will be the same proof except where we replace every instance of ψ with $\neg\psi$. We assume that m_Φ is ψ-sufficient and not $\neg\psi$-sufficient in S, and show that this leads to a contradiction. From the assumption that m_Φ is not $\neg\psi$-sufficient, we have that $S, r \vDash \psi$ and $S, m_\Phi(r) \nvDash \psi$ for a given $r \in \mathcal{R}_S$ (note that we omit the case

when $S, r \not\models \psi$ and $S, m_\Phi(r) \models \psi$ for a given $r \in \mathcal{R}_S$, since this would imply that $r \notin m_\Phi(r)$ which is an immediate contradiction). From $S, m_\Phi(r) \not\models \psi$ we have that $S, r' \models \neg\psi$ for a given $r' \in m_\Phi(r)$. Due to the fact that m_Φ is ψ-sufficient, we can conclude that $S, m_\Phi(r') \models \neg\psi$. However, because m_Φ is an equivalence classifier, we have that $r \in m_\Phi(r')$, and since $S, r \models \psi$, we also have that $S, m_\Phi(r') \not\models \neg\psi$. Contradiction! □

Although the above result is established for LTL-based monitors, in the general case it holds for every monitor with the *symmetric property*, i.e. it holds for every monitor m for which we have that $r \in m(r')$ implies $r' \in m(r)$ (and vice-versa). Using this result, we can establish our first characterization of when an LTL-based monitor is sufficient. Even though this characterization might not immediately appear useful from a computational perspective, we can use it afterwards to establish a correspondence between sufficient and redundant monitors.

Proposition 6.8 (Characterization of LTL-monitors)**.** *Given a transition system S, a monitor m_Φ, where Φ is a non-empty finite set of formulas, and an LTL-based norm ψ. We have:*

$$m_\Phi \text{ is } \psi\text{-sufficient in } S \quad \Leftrightarrow \quad \forall r \in \mathcal{R}_S : S, m_\Phi(r) \models \psi \text{ or } S, m_\Phi(r) \models \neg\psi$$

Proof. We show the direction from left to right ($\Rightarrow$) and from right to left ($\Leftarrow$) separately.

- $\Rightarrow$ Suppose m_Φ is ψ-sufficient. By Proposition 6.7 we have that m_Φ is also $\neg\psi$-sufficient. Now consider an arbitrary $r \in \mathcal{R}_S$. If it is the case that $S, r \models \neg\psi$, then by ψ-sufficiency we have that $S, m(r) \models \neg\psi$. Otherwise, if $S, r \models \psi$, then by $\neg\psi$-sufficiency we have that $S, m_\Phi(r) \models \psi$. Since r was chosen arbitrarily, we have that $\forall r \in \mathcal{R}_S$ it either holds that $S, m_\Phi(r) \models \psi$ or that $S, m_\Phi(r) \models \neg\psi$.

- $\Leftarrow$ Suppose m_Φ is not ψ-sufficient. From this we can conclude that $S, r \models \neg\psi$ and $S, m_\Phi(r) \not\models \neg\psi$ for a certain $r \in \mathcal{R}_S$. By Proposition 6.6 we have that m_Φ is correct, and thus that $r \in m_\Phi(r)$, implying that also $S, m_\Phi(r) \not\models \psi$. Thus, we have that $\exists r \in \mathcal{R}_S$ such that $S, m_\Phi \not\models \psi$ and $S, m_\Phi \not\models \neg\psi$, as needed.

□

Although this characterization is not immediately useful in order to decide whether a monitor is sufficient from a computational perspective (since the set of runs $\mathcal{R}_S$ for a given transition system S can be infinitely large), we can use it to establish the next result. This result shows that deciding sufficiency or redundancy of a monitor are the same problems.

Theorem 6.1 (Correspondence between sufficiency and redundancy)**.** *Given a transition system S, a monitor m_Φ, where Φ is a non-empty finite set of LTL formulas, and an LTL formula φ. We have:*

$$m_\varphi \text{ is redundant for } m_\Phi \text{ in } S \quad \Leftrightarrow \quad m_\Phi \text{ is } \varphi\text{-sufficient in } S$$

Proof. We use the result acquired from Proposition 6.8. From left to right and from right to left:

$$\begin{aligned}
&m_\Phi \text{ is } \varphi\text{-sufficient in } S \quad \Leftrightarrow \\
\forall r \in \mathcal{R}_S : \;& m_\Phi(r) \vDash \varphi \text{ or } m_\Phi(r) \vDash \neg\varphi \quad \Leftrightarrow \\
\forall r \in \mathcal{R}_S : \;& m_\Phi(r) \subseteq m_\varphi(r) \quad \Leftrightarrow \\
\forall r \in \mathcal{R}_S : \;& m_\Phi(r) \cap m_\varphi(r) = m_\Phi(r) \quad \Leftrightarrow \\
\forall r \in \mathcal{R}_S : \;& m_{\Phi\cup\{\varphi\}}(r) = m_\Phi(r) \quad \Leftrightarrow \\
m_{\Phi\cup\{\varphi\}} = m_\Phi \quad \Leftrightarrow \quad & m_\varphi \text{ is redundant for } m_\Phi \text{ in } S
\end{aligned}$$

□

This Theorem implies that if we would solve one of the problems (either sufficiency or redundancy), we automatically solve the other problem. Now that we have established this result, we will focus our attention completely on the problem of deciding whether a monitor is sufficient or not, and by doing so we inherently solve the problem of determining redundancy.

In order to decide whether a monitor is sufficient for a given norm, it is important to see that LTL-based monitors are already ψ-sound for any LTL-norm ψ. We can establish this result from the fact that each LTL-based monitor is an equivalence classifier.

Proposition 6.9. *Let ψ be an LTL-norm and S be a transition system. The combined monitor m_Φ is ψ-sound in S.*

Proof. We assume the contrary. That is, there is a run $r \in \mathcal{R}_S$ such that $S, m_\Phi(r) \vDash \neg\psi$ and $S, r \vDash \psi$. By Proposition 6.6 we have that m_Φ is correct, and thus that $r \in m_\varphi(r)$ and hence $S, r \vDash \neg\psi$. Contradiction! □

Although this result is established for LTL-based monitors, in the general case it holds for every monitor that is *correct*. An important consequence of this proposition is that in order to show that an LTL monitor is ψ-sufficient, we only have to show that this monitor is ψ-complete, since this monitor is already ψ-sound. From Proposition 6.6 we have already shown that for a given LTL-monitor m_Φ, where Φ is a finite set of LTL-formulas, there is a direct correspondence between an equivalence class from the set $\mathcal{C}(m_\Phi)$ and a formula $\hat{\varphi} \in \mathsf{Con}(\Phi)$. Particularly, each element in $\mathcal{C}(m_\Phi)$ has a unique

representative in $\mathsf{Con}(\Phi)$, which we will prove in the upcoming Lemma. This Lemma will play crucial role in order to prove Theorem 6.2 and Theorem 6.3 (and Proposition 6.10 and Proposition 6.11).

Lemma 6.1. *Given a transition system S. We have that for each run $r \in \mathcal{R}_S$ there is a unique $\hat{\varphi} \in \mathsf{Con}(\Phi)$ with $S, r \vDash \hat{\varphi}$. We denote this formula by $\hat{\varphi}(r)$.*

Proof. Clearly, for each $\varphi \in \Phi$ either $S, r \vDash \varphi$ or $S, r \vDash \neg\varphi$. Then, $\hat{\varphi}(r) = \varphi'_1 \wedge \cdots \wedge \varphi'_n$ with $\varphi'_i \in \{\varphi_i, \neg\varphi_i\}$, and where $\varphi'_i = \varphi_i$ if and only if $S, r \vDash \varphi_i$. It is also clear that there can be no other formula $\hat{\varphi}' \in \mathsf{Con}(\Phi)$ with $S, r \vDash \hat{\varphi}'$. □

Finally, we turn to our main results about LTL-monitors. Theorem 6.2 gives us a way in which we can logically characterize when a monitor is sufficient for a given norm and transition system. In particular, given a monitor m_Φ and LTL-norm ψ, it says that m_Φ is ψ-sufficient if and only if there is a subset of $\mathsf{Con}(\Phi)$ such that the disjunction of this subset is logically equivalent to $\neg\psi$. Theorem 6.3 gives us a way in which we can devise a decision procedure in order to verify whether a monitor is ψ-sufficient, which will play an important role when we want to consider the complexity of model checking.

Theorem 6.2 (Characterization of LTL-monitors)**.** *Let $\Phi = \{\varphi_1, \ldots, \varphi_n\}$ be a non-empty finite set of LTL-formulas, ψ be an LTL-norm, and S be a transition system. Then m_Φ is ψ-sufficient in S if and only if:*

$$S \vDash \neg\psi \rightarrow \bigvee\{\hat{\varphi} \in \mathsf{Con}(\Phi) \mid S \vDash \hat{\varphi} \rightarrow \neg\psi\}$$

Proof. We show the direction from left to right ($\Rightarrow$) and from right to left ($\Leftarrow$) separately.

- $\Rightarrow$ Suppose $S \nvDash \neg\psi \rightarrow \bigvee\{\hat{\varphi} \in \mathsf{Con}(\Phi) \mid S \vDash \hat{\varphi} \rightarrow \neg\psi\}$, we show that this implies that m_Φ is not ψ-sufficient in S. From our assumption, we have that $S, r \vDash \neg\psi$ and $S, r \nvDash \bigvee\{\hat{\varphi} \in \mathsf{Con}(\Phi) \mid S \vDash \hat{\varphi} \rightarrow \neg\psi\}$ for some $r \in \mathcal{R}_S$. We invoke Lemma 6.1 to retrieve $\hat{\varphi}(r)$, for which we have that $S, r \vDash \hat{\varphi}(r)$. But, since $S, r \nvDash \bigvee\{\hat{\varphi} \in \mathsf{Con}(\Phi) \mid S \vDash \hat{\varphi} \rightarrow \neg\psi\}$, it must be that $\hat{\varphi}(r) \notin \{\hat{\varphi} \in \mathsf{Con}(\Phi) \mid S \vDash \hat{\varphi} \rightarrow \neg\psi\}$, implying that $S \nvDash \hat{\varphi}(r) \rightarrow \neg\psi$. This implies that $S, r' \vDash \hat{\varphi}(r) \wedge \psi$ for some $r' \in \mathcal{R}_S$, which implies that $r' \in m_\Phi(r)$ and thus $S, m_\Phi(r) \nvDash \neg\psi$. But since $S, r \vDash \neg\psi$, we have shown that m_Φ is not ψ-sufficient in S, as needed.

- $\Leftarrow$ Assume $S \vDash \neg\psi \rightarrow \bigvee\{\hat{\varphi} \in \mathsf{Con}(\Phi) \mid S \vDash \hat{\varphi} \rightarrow \neg\psi\}$, and assume an arbitrary $r \in \mathcal{R}_S$ for which $S, r \vDash \neg\psi$. From this, we can conclude that $S, r \vDash \bigvee\{\hat{\varphi} \in \mathsf{Con}(\Phi) \mid S \vDash \hat{\varphi} \rightarrow \neg\psi\}$. We invoke Lemma 6.1 to retrieve $\hat{\varphi}(r)$, for which we have that $S, r \vDash \hat{\varphi}(r)$, allowing us to conclude that $\hat{\varphi}(r) \in \{\hat{\varphi} \in \mathsf{Con}(\Phi) \mid S \vDash \hat{\varphi} \rightarrow \neg\psi\}$. From this, we can conclude that $S \vDash \hat{\varphi}(r) \rightarrow \neg\psi$,

which in turn implies that $S, m_\Phi(r) \vDash \neg\psi$. Since r was chosen arbitrarily, we may conclude that this holds for every $r \in \mathcal{R}_S$ for which $S, r \vDash \neg\psi$, which shows that m_Φ is ψ-sufficient in S.

□

This theorem describes the general case for any combined or non-combined LTL-based monitor. In the next proposition we apply this theorem to the special case of a non-combined monitor.

Proposition 6.10 (Characterization of non-combined LTL-monitors)**.** *Let m_φ be a non-combined LTL monitor, ψ an LTL-norm, and S a transition system. Then m_φ is ψ-sufficient in S if and only if:*

$$S \vDash \neg\psi \quad or \quad S \vDash \psi \quad or \quad S \vDash \neg\psi \leftrightarrow \varphi \quad or \quad S \vDash \psi \leftrightarrow \varphi$$

Proof. This is a special case of Theorem 6.2 where $\Phi = \{\varphi\}$ and thus $\mathsf{Con}(\Phi) = \{\varphi, \neg\varphi\}$. We consider the following cases:

- If $\{\varphi, \neg\varphi\} = \{\hat{\varphi} \in \mathsf{Con}(\Phi) \mid S \vDash \hat{\varphi} \to \neg\psi\}$, i.e. $S \vDash (\varphi \to \psi) \wedge (\neg\varphi \to \psi)$, it must be the case that $S \vDash \neg\psi$. Since $S \vDash \neg\psi \to \top$ is a tautology, we only need to verify whether $S \vDash \neg\psi$.
- If $\varnothing = \{\hat{\varphi} \in \mathsf{Con}(\Phi) \mid S \vDash \hat{\varphi} \to \neg\psi\}$, then $S \vDash \neg\psi \to \bigvee\{\hat{\varphi} \mid \hat{\varphi} \in \mathsf{Con}(\Phi) \text{ and } S \vDash \hat{\varphi} \to \neg\psi\}$ reduces to $S \vDash \neg\psi \to \bot$, which is equivalent to $S \vDash \psi$.
- If $\{\varphi\} = \{\hat{\varphi} \in \mathsf{Con}(\Phi) \mid S \vDash \hat{\varphi} \to \neg\psi\}$, i.e. $S \vDash \varphi \to \neg\psi$, and we must check that $S \vDash \neg\psi \to \phi$, then the combined formula that we need to check is $S \vDash \neg\psi \leftrightarrow \varphi$.
- If $\{\neg\varphi\} = \{\hat{\varphi} \in \mathsf{Con}(\Phi) \mid S \vDash \hat{\varphi} \to \neg\psi\}$, i.e. $S \vDash \neg\varphi \to \neg\psi$, and we must check that $S \vDash \neg\psi \to \neg\phi$, then the combined formula that we need to check is $S \vDash \psi \leftrightarrow \varphi$.

□

Thus, this proposition entails that for a given transition system S, non-combined LTL monitor m_φ and LTL-norm ψ, in order to verify whether m_φ is ψ-sufficient in S, we have to check the validity of at most 4 distinct LTL-formulas in S. Thus, it is already apparent that the complexity for checking whether a *non-combined* monitor is sufficient in a given transition system remains close to the complexity of LTL model checking. However, in the general case, that is to say the result that have acquired from Theorem 6.2, we have that in order to verify whether a combined monitor m_Φ is ψ-sufficient in S,

we have to check the validity of $O(2^{|\Phi|})$ number of formulas. Thus, it is not at all apparent whether the complexity for checking whether a *combined* monitor is sufficient in a given transition system remains close to the complexity of LTL model checking as well. This leads us to the next theorem, which, as we will see in the next section, allows us to devise a generic decision procedure in order to verify whether a monitor is sufficient in a given transition system.

Theorem 6.3 (Decision procedure for LTL-monitors)**.** *Let $\Phi = \{\varphi_1, \ldots, \varphi_n\}$ be a non-empty finite set of LTL-formulas, ψ be an LTL-norm, and S be a transition system. Then, m_Φ is ψ-sufficient in S if and only if for all $\hat{\varphi} \in \mathsf{Con}(\Phi)$:*

$$\neg\psi \wedge \hat{\varphi} \text{ is satisfiable in } S \quad \Rightarrow \quad S \vDash \hat{\varphi} \rightarrow \neg\psi$$

Proof. We show the direction from left to right ($\Rightarrow$) and from right to left ($\Leftarrow$) separately.

- $\Rightarrow$ Suppose m_Φ is ψ-sufficient in S, that is, $\forall r \in \mathcal{R}_S$ we have that $S, r \vDash \neg\psi$ implies $S, m_\Phi(r) \vDash \neg\psi$. Suppose an arbitrary $\hat{\varphi} \in \mathsf{Con}(\Phi)$ for which $\neg\psi \wedge \hat{\varphi}$ is satisfiable in S. This means that $S, r \vDash \neg\psi \wedge \hat{\varphi}$ for some $r \in \mathcal{R}_S$. Moreover, assume that $S \nvDash \hat{\varphi} \rightarrow \neg\psi$, that is, $S, r' \vDash \hat{\varphi} \wedge \psi$ for some $r' \in \mathcal{R}_S$. If we can derive a contradiction, it must hold that $S \vDash \hat{\varphi} \rightarrow \neg\psi$, as needed. However, since $r \vDash \hat{\varphi}$ and $r' \vDash \hat{\varphi}$, we have that $r' \in m_\Phi(r)$, implying $S, m_\Phi(r) \nvDash \neg\psi$ and thus $S, r \nvDash \neg\psi$. Contradiction!

- $\Leftarrow$ We assume that m_Φ is not ψ-sufficient in S, and show that this implies the existence of a $\hat{\varphi} \in \mathsf{Con}(\Phi)$ such that $\neg\psi \wedge \hat{\varphi}$ is satisfiable in S and $S \nvDash \hat{\varphi} \rightarrow \neg\psi$. From our assumption, we have that $S, r \vDash \neg\psi$ and $S, m_\Phi(r) \nvDash \neg\psi$ for some $r \in \mathcal{R}_S$. The latter implies that $S, r' \vDash \psi$ for some $r' \in m_\Phi(r)$. We now invoke Lemma 6.1 to retrieve $\hat{\varphi}(r')$, for which we have that $S, r \vDash \neg\psi \wedge \hat{\varphi}(r')$, i.e. $\neg\psi \wedge \hat{\varphi}(r')$ is satisfiable in S. Since we also have that $S, r' \vDash \hat{\varphi}(r') \wedge \psi$, we can conclude $S \nvDash \hat{\varphi}(r') \rightarrow \neg\psi$, as needed.

□

Thus, both Theorem 6.2 and Theorem 6.3 give us a way to reduce the problem of checking whether an LTL-based monitor is sufficient for a given norm to a problem of LTL model checking. Although both these theorems show different results, they are equally suitable to prove sufficiency of a monitor. In the next proposition we again apply our last theorem to the special case of a non-combined monitor.

Proposition 6.11 (Decision procedure for non-combined LTL-monitors)**.** *Let m_φ be a non-combined LTL monitor, ψ an LTL-norm, and S a transition*

system. Then m_φ is ψ-sufficient in S if and only if:

$$\begin{aligned} \neg\psi \wedge \varphi \text{ is satisfiable in } S &\Rightarrow S \vDash \varphi \rightarrow \neg\psi \\ \neg\psi \wedge \neg\varphi \text{ is satisfiable in } S &\Rightarrow S \vDash \neg\varphi \rightarrow \neg\psi \end{aligned}$$

Proof. This follows immediately from the special case of Theorem 6.3 where $\Phi = \{\varphi\}$ and thus $\mathsf{Con}(\Phi) = \{\varphi, \neg\varphi\}$. □

We now have developed sufficient tools to answer several questions related to model checking complexity of LTL-based monitors, which will be the topic of the next section.

6.6 Model checking

The model checking problems we are interested in are not only related to the verification of sufficient monitors, but also in determining whether we can construct or find such a monitor. For a given transition system S we consider the following list of problems:

1. Given an LTL-norm ψ, does there exist an LTL-monitor m_φ which is ψ-sufficient in S?
2. Given an LTL-norm ψ ...
   (a) ... and a non-combined monitor m_φ, is m_φ ψ-sufficient in S?
   (b) ... and a combined monitor m_Φ, is m_Φ ψ-sufficient in S?
3. Given an LTL-norm ψ and a finite set of non-combined LTL-monitors M ...
   (a) ... is there a non-combined monitor $m_\varphi \in M$ which is ψ-sufficient in S?
   (b) ... is there a combined monitor $m_{\varphi_1} \oplus \ldots \oplus m_{\varphi_k}$ $(m_{\varphi_1}, \ldots, m_{\varphi_k} \in M)$ which is ψ-sufficient in S?

We can immediately answer the first question, since it is easy to show that we can always construct an LTL-based monitor which is sufficient for a given LTL-norm and transition system.

Proposition 6.12 (Problem 1)**.** *Given a transition system S, we have that for every LTL-norm ψ there exists an LTL-monitor m_φ such that m_φ is ψ-sufficient.*

Proof. This trivially holds by using Proposition 6.10. This proposition tells us that monitor m_ψ is ψ-sufficient for any transition system S. □

For the upcoming result, we use several theorems and propositions from the previous section. We define the size of a transition system as the number of states and transitions in said system. We have the following result for non-combined monitors.

Proposition 6.13 (Problem 2(a))**.** *Let S be a transition system, m_φ a non-combined LTL-monitor, and ψ an LTL-norm. The problem whether m_φ is ψ-sufficient in S is* **PSPACE***-complete in the size of S, the length of φ and the length of ψ.*

Proof. Membership directly follows from Proposition 6.10, where we have to check the validity of at most four LTL-formulas, which can be checked in **PSPACE** by using standard LTL model checking, as shown in the work of [Sistla and Clarke, 1985]. To achieve the lower bound of **PSPACE**-hardness, we reduce LTL model checking, a **PSPACE**-complete problem, to our problem. Let S be an arbitrary transition system and φ an arbitrary LTL-formula. LTL model checking asks whether $S \vDash_{\text{LTL}} \varphi$. Our reduction consists of the following steps:

1. We introduce a fresh proposition p neither occurring in S nor in φ, and let S_p be an exact copy of S, but for which p is added to the valuation of the initial state. We can establish the following:

$$\begin{array}{lll} S \vDash \varphi & \Leftrightarrow & S_p \vDash \varphi \\ S \nvDash p & \text{and} & S_p \vDash p \end{array}$$

2. Let S' be the disjoint union of S and S_p, together with a new initial state q_0' which has an outgoing edge to the initial state of S and an outgoing edge to the initial state of S_p.

3. We let the LTL-norm ψ be defined as $\bigcirc(\varphi \vee p)$, and we consider the monitor $m_\top$. By Proposition 6.10, we have that $m_\top$ is ψ-sufficient in S' iff $S' \vDash \neg\psi$, or $S' \vDash \psi$, or $S' \vDash \neg\psi \leftrightarrow \top$, or $S' \vDash \psi \leftrightarrow \top$, which reduces to $S' \vDash \neg\psi$, or $S' \vDash \psi$, i.e. $S' \vDash \neg\bigcirc(\varphi \vee p)$, or $S' \vDash \bigcirc(\varphi \vee p)$. However, we have that $S' \nvDash \neg\bigcirc(\varphi \vee p)$, since $S' \nvDash \bigcirc(\neg\varphi \wedge \neg p)$, since $S' \nvDash \bigcirc\neg p$, since $S_p \vDash p$. Hence, we must show that $S' \vDash \bigcirc(\varphi \vee p)$.

By construction of S', we have:

$$S' \vDash \bigcirc(\varphi \vee p) \;\Leftrightarrow\; S' \vDash \bigcirc\varphi \;\Leftrightarrow\; S \vDash \varphi$$

This gives the following polynomial-time reduction:

$$S \vDash_{\text{LTL}} \varphi \;\Leftrightarrow\; m_\top \text{ is } \bigcirc(\varphi \vee p)\text{-sufficient in } S'$$

□

For combined monitors, we have a similar result, which at first glance may seem surprising. However, the reason for this result is the fact that complexity class of polynomial space is closed under non-determinism. We have the following important result for combined monitors.

Proposition 6.14 (Problem 2(b))**.** *Let S be a transition system, m_Φ a combined LTL-monitor, and ψ an LTL-norm. The problem whether m_Φ is ψ-sufficient in S is* **PSPACE***-complete in the size of S, the size of Φ, and the length of ψ.*

Proof. To show that the problem is in **PSPACE** we use Theorem 6.3. The following procedure runs in **NPSPACE**, i.e. non-deterministic polynomial space. This is sufficient for the result, since **PSPACE** is closed under non-determinism as shown in [Savitch, 1970]. We guess a formula $\hat{\varphi} \in \mathsf{Con}(\Phi)$ and check whether $\neg\psi \wedge \hat{\varphi}$ is satisfiable (this can be done in polynomial space by checking $S \not\models \neg(\neg\psi \wedge \hat{\varphi})$) and whether $S \not\models \hat{\varphi} \rightarrow \neg\psi$ (again, this can be done in polynomial space). If the answer is yes, m_Φ is not ψ-sufficient in S. Since **PSPACE** is closed under complement, we establish our result. Hardness transfers from Proposition 6.13 where we can perform the same polynomial-time reduction with $\Phi = \{\top\}$. □

The third problem is a straightforward extension of the second. We have the following result for non-combined monitors.

Proposition 6.15 (Problem 3(a))**.** *Let S be a transition system, M a finite set of non-combined LTL-monitors, and ψ an LTL-norm. The problem whether there exists a monitor $m_\varphi \in M$ which is ψ-sufficient in S is* **PSPACE***-complete in the size of S, the size of M, and the length of ψ.*

Proof. Membership follows from the fact that it is sufficient to call the membership procedure from Proposition 6.13 at most $|M|$-times. Hardness transfers from hardness of Proposition 6.13, where we are able to perform the same polynomial-time reduction with $M = \{m_\top\}$. □

Moreover, we have the following result for combined monitors.

Proposition 6.16 (Problem 3(b))**.** *Let S be a transition system, M a finite set of non-combined LTL-monitors, and ψ an LTL-norm. The problem whether there is a combined monitor $m_1 \oplus \ldots \oplus m_k$ $(m_{\varphi_1}, \ldots, m_{\varphi_k} \in M)$ which is ψ-sufficient in S is* **PSPACE***-complete in the size of S, the size of M, and the length of ψ.*

Proof. The following procedure runs in **NPSPACE**, i.e. non-deterministic polynomial space, which is again sufficient for the result, since **PSPACE** is

closed under non-determinism as shown in [Savitch, 1970]. We guess a non-empty subset $\{m_{\varphi_1}, \ldots, m_{\varphi_k}\} \subseteq M$ and perform the membership procedure described in Proposition 6.14 for $\Phi = \{\varphi_1, \ldots, \varphi_k\}$. Hardness transfers from hardness of Proposition 6.13, where we can perform the same polynomial-time reduction with $M = \{m_\top\}$. □

This concludes the model checking problems that we wanted to consider in this chapter. We note however that the list of problems that we supplied in the beginning of this section is by no means *exhaustive*; certainly more issues related to the verification of monitors can be considered which lay beyond the scope of this chapter.

6.7 Discussion

In this chapter we developed a basic framework in which monitors can be studied and verified. Formal verification of monitors is of crucial importance to determine whether a normative multi-agent system meets its design objectives. The verification problem we addressed is how to verify whether the monitors are sufficient for the norm, that is, verify whether the monitors correctly and completely detect all the possible violations that can occur. We proposed different types of monitors, provided a logical analysis of monitors, studied the relations between monitors and norms to be monitored, and explored computational aspects of verifying whether a monitor is sufficient for a normative multi-agent system. The language on which these monitors and norms were built was LTL. As it turned out, combining these simple monitors allowed us to construct complex monitors with vastly growing reasoning capabilities. Moreover, showing that such a monitor is sufficient for a given norm (a violation is always correctly detected) still lies within the complexity bounds of LTL model checking itself.

This chapter can be extended in multiple ways, which we will briefly reflect upon. First of all, it is important to consider monitors which are not fully specified by some function, but for which only a partial specification is given. It can be interesting to consider more classes of monitors, for example non-binary or non-correct monitors. With respect to the latter, an interesting study would be to use our framework to detect faulty monitors within a system. Dynamic monitors, i.e. monitors that can change their observations over time such as a moving camera, are also important to study. Moreover, it is important to add *costs* to the framework and to study optimality properties. Finally, we would like to note that although our setting is defined over infinite runs it can also be given over finite runs. In the case of LTL monitors one can use a finite-trace semantics; in particular, a three-valued semantics similar to [Bauer

et al., 2011] would be interesting in the context of monitoring norm violations. We leave a detailed study for future research.

Chapter 7

Discussion

In this chapter we reflect upon the work done in this thesis, and argue how this work contributes to the field of normative multi-agent systems, and Artificial Intelligence in general. Moreover, we want to briefly explore several possibilities for future work based on the contributions of this thesis.

7.1 Conclusions

In this section we return to our initial research questions, and recapitulate how the work done in this thesis can be used to answer these questions. This thesis studied enforcement norms, which are norms for which agents can obey or violate, and which can lead to sanctions when a violation is detected. Moreover, we consider multi-agent systems in which the specification and internal architecture of the participating agents are primarily unknown to us. This implied that we could not simply assume that the agents are aware of the norms, or that they are compliant with respect to the norms.

In Research question 1, we asked ourselves how we can design and analyse logic-based frameworks which are able to capture the intricate interplay between logic, norms and games in order to model and verify normative multi-agent systems. In Chapter 3 and Chapter 4, we developed separate logical frameworks which allowed us to answer this question. In Chapter 3 we developed a logic that allows reasoning of agents' strategic abilities under a multitude of compliance types. In this approach, the designer attributes compliance types to (coalitions of) agents, allowing us to verify whether the design objectives are satisfied in the system. This extension of Alternating-time Temporal Logic enables us to reason about the abilities of (coalitions of) agents under various compliance types. We showed that the model checking complexity of this logic is equivalent to the model checking complexity of standard Alternating-time Temporal Logic. In Chapter 4, we developed a logical proof

system that allows us to specify and verify various implementability questions related to normative multi-agent systems. The idea is that by attributing types to agents, we were able to analyse the normative multi-agent system using game theoretic solution concepts. Since norms, once implemented, can change the environment of the system, these norms can act as a *mechanism* that can change the underlying game. Using this framework, we answered the research question using principles from *mechanism design*. The two distinctly different approaches found in Chapter 3 and Chapter 4 highlights that there is not one unique way of answering this research question.

In Research question 2, we asked ourselves how can we design and analyse a logic-based framework to model and verify normative multi-agent systems situated in a dynamic environment. In Chapter 5, we developed a dynamic logic and accompanying proof system that allowed us to answer this question. The view we adopt in this chapter is that (normative) multi-agent systems can be modelled by pointed labelled transition systems, which show which (normative) facts become true under execution of which actions. Adding a norm is an operation on these models: they transform normative multi-agent systems such that the resulting system is aligned with the added norm. In this chapter we explored how these operations work, and how we can use these operations in combination with our logic to prove various properties of a normative multi-agent system in a dynamic environment.

Finally, in Research question 3, we asked ourselves how can we design and analyse a logic-based framework to model and verify monitors observing the behaviour of a normative multi-agent system. In Chapter 6, we developed such a framework in which monitors can be studied and verified. We provided a logical analysis of monitors, studied the relations between monitors and norms to be monitored, and explored computational aspects of verifying whether a monitor is sufficient for a normative multi-agent system.

7.2 Contributions

The contributions of this thesis are both theoretical and practical in nature. From a theoretical perspective, the contributions in Chapter 3, 4, 5 and 6 can be summarized as follows:

1. In each chapter, we provide a formal framework in which the corresponding verification problems can be studied. This includes a formal specification of the multi-agent system and the relevant norms we want to study.

2. In each chapter, we introduce or extend a formal logic in which our verification problems can be expressed. We provide the corresponding

syntax and semantics, and show how these logics allow us frame the verification problems.

3. In each chapter, we analyse our logic, and provide relevant properties in propositions, lemma's and theorems.

4. In each chapter, we provide an accompanying proof systems and/or provide accompanying model checking results, allowing us to give several theoretical results of these logics.

As such, the theoretical contribution of this thesis lies in the fact that we bridge the gap between the field of *normative multi-agent systems* and *the modelling and verification of formal systems*.

From a practical perspective, this thesis sets a foundation for the development of tools in which normative multi-agent systems can be verified. As we already stated in Chapter 1, normative systems are making their way into our everyday life, for example electronic marketplaces where agents can buy and sell goods, or smart grid energy systems where agents can exchange energy based on supply and demand. Formal verification is of crucial importance if we are looking for a *guarantee* that the system is correct. Whenever the cost of defection is high, it is of crucial importance that we know that a system is correct without actually having to run it. In an electronic marketplace, defection can imply a loss of money, and in a smart road system defection can even imply physical injury. Verification gives us this guarantee. Development of such methods and tools play an important role in the advancement of normative multi-agent systems and Artificial Intelligence in general.

It is however important to note that this thesis does not settle the modelling and verification issue of normative multi-agent systems in general. For example, one of the topics that we do not cover in this thesis is how sanctioning works, and how possible sanctioning mechanisms can be implemented. A verification problem related to this is to determine whether a sanctioning mechanism is correct with respect to the given norms in the multi-agent system.

For future work, a possibility is to extend the work presented in this thesis, or to study related problems. An important extension of this work is to study the specific algorithms needed to perform the verification task. Currently, we were merely concerned with the frameworks in which these verification problems can be studied. In this thesis we analysed multi-agent systems with slightly varying characteristics across the chapters (although we always stayed close to the generic model of transition systems), and analysed various different classes of norms. A way to extend the work in this thesis is thus to look for a unifying framework in which the verification problems can be analysed.

A crucial related problem of verification is *synthesis* and *design*. Verification only gives us a yes/no answer, but design allows us to *find* these good solutions. This thesis did not cover any design and synthesis questions, but serves as a stepping stone towards these questions. Moreover, as we already mentioned previously, there exists many verification questions a designer might ask which are not covered by this thesis. Thus, there is a need to address additional verification questions, for example related to that of enforcement mechanisms. Additionally, we can consider other types of norms, can consider other means of testing a system, for example a validation procedure, can consider other kinds of verification, such as run-time verification, and can consider other kinds of multi-agent systems.

Bibliography

Ågotnes, T., van der Hoek, W., and Wooldridge, M. (2007). Normative system games. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '07, pages 881–888.

Ågotnes, T., van der Hoek, W., and Wooldridge, M. (2010). Robust normative systems and a logic of norm compliance. *Logic Journal of the IGPL*, 18(1).

Alchourrón, C. E. and Bulygin, E. (1971). *Normative systems*. Library of exact philosophy. Springer-Verlag.

Alchourrón, C. E., Gärdenfors, P., and Makinson, D. (1985). On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50(2):510–530.

Alechina, N., Bulling, N., Dastani, M., and Logan, B. (2015). Practical runtime norm enforcement with bounded lookahead. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, pages 443–451. IFAAMAS.

Alechina, N., Dastani, M., and Logan, B. (2013). Reasoning about normative update. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 20–26.

Alur, R., Henzinger, T. A., and Kupferman, O. (2002). Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713.

Anderson, A. R. (1958). A reduction of deontic logic to alethic modal logic. *Mind*, 67(265):100–103.

Arrow, K. J. (1950). A difficulty in the concept of social welfare. *The Journal of Political Economy*, 58(4):328–346.

Arrow, K. J. (1951). *Social Choice and Individual Values*. New Haven.

Artikis, A. and Sergot, M. (2010). Executable specification of open multi-agent systems. *Logic Journal of IGPL*, 18(1):31–65.

Aucher, G., Grossi, D., Herzig, A., and Lorini, E. (2009). Dynamic context logic. In *Proceedings of Logic, Rationality and Interaction*, LORI '09, pages 15–26.

Baier, C. and Katoen, J.-P. (2008). *Principles of Model Checking (Representation and Mind Series)*. The MIT Press.

Bauer, A., Leucker, M., and Schallhart, C. (2011). Runtime verification for LTL and TLTL. *ACM Transactions on Software Engineering and Methodology*, 20(4):14:1–14:64.

Beeri, C. (1980). On the menbership problem for functional and multivalued dependencies in relational databases. *ACM Transactions on Database Systems*, 5(3):241–259.

Boella, G., Pigozzi, G., and van der Torre, L. (2009). Normative framework for normative system change. In *Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '09, pages 169–176.

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

Bratman, M. (1987). *Intention, plans, and practical reason*. Harvard University Press.

Broersen, J. (2004). Action negation and alternative reductions for dynamic deontic logics. *Journal of Applied Logic*, 2(1):153–168.

Bulling, N. and Dastani, M. (2011a). Normative programs and normative mechanism design. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 3*, AAMAS '11, pages 1187–1188. International Foundation for Autonomous Agents and Multiagent Systems.

Bulling, N. and Dastani, M. (2011b). Verifying normative behaviour via normative mechanism design. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, IJCAI '11, pages 103–108.

Bulling, N., Dastani, M., and Knobbout, M. (2013). Monitoring norm violations in multi-agent systems. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 491–498. IFAAMAS.

Bulling, N., Jamroga, W., and Dix, J. (2009). Reasoning about temporal properties of rational play. *Annals of Mathematics and Artificial Intelligence*, 53(1):51–114.

Chandy, K. M. and Misra, J. (1989). *Parallel Program Design - A Foundation*. Addison-Wesley Publishing Company.

Chellas, B. F. (1980). *Modal Logic: An Introduction*. Cambridge University Press.

Clarke, E. M. and Emerson, E. A. (1982). Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs, Workshop*, pages 52–71.

Clarke, E. M., Grumberg, O., and Peled, D. (2001). *Model checking*. MIT Press.

Cohen, P. R. and Levesque, H. J. (1990). Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261.

Dastani, M., Grossi, D., Meyer, J.-J. C., and Tinnemeier, N. A. M. (2009a). Normative multi-agent programs and their logics. In *Knowledge Representation for Agents and Multi-Agent Systems*, pages 16–31. Springer.

Dastani, M., Meyer, J.-J. C., and Grossi, D. (2011). A logic for normative multi-agent programs. *Journal of Logic and Computation*.

Dastani, M., Tinnemeier, N. A. M., and Meyer, J.-J. C. (2009b). A programming language for normative multi-agent systems. *Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, pages 397–417.

Dennett, D. (1987). *The Intentional Stance*. MIT Press.

Fikes, R. E. and Nilsson, N. J. (1971). Strips: A new approach to the application of theorem proving to problem solving. In *Proceedings of the 2nd International Joint Conference on Artificial Intelligence*, IJCAI '71, pages 608–620. Morgan Kaufmann Publishers Inc.

Fischer, M. J. and Ladner, R. E. (1977). Propositional modal logic of programs. In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*, pages 286–294.

Grossi, D. (2007). *Designing invisible handcuffs: Formal investigations in institutions and organizations for multi-agent systems*. PhD thesis, SIKS Dissertation Series.

Grossi, D., Aldewereld, H., and Dignum, F. (2007). Ubi lex, ibi poena: Designing norm enforcement in e-institutions. In *Coordination, Organizations, Institutions, and Norms in Agent Systems II*, volume 4386 of *Lecture Notes in Computer Science*, pages 101–114. Springer Berlin Heidelberg.

Huth, M. and Ryan, M. (2004). *Logic in Computer Science: Modelling and Reasoning About Systems*. Cambridge University Press.

Jamroga, W., Hoek, W., and Wooldridge, M. (2005). *Progress in Artificial Intelligence: 12th Portuguese Conference on Artificial Intelligence, EPIA 2005, Covilhã, Portugal, December 5-8, 2005. Proceedings*, chapter Intentions and Strategies in Game-Like Scenarios, pages 512–523. Springer Berlin Heidelberg.

Knobbout, M. and Dastani, M. (2012). Reasoning under compliance assumptions in normative multiagent systems. In *Proceedings of the 11th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '12, pages 331–340.

Knobbout, M., Dastani, M., and Meyer, J.-J. C. (2014). Reasoning about dynamic normative systems. In *Logics in Artificial Intelligence - 14th European Conference*, JELIA '14, pages 628–636.

Lipeck, U. W. and Saake, G. (1987). Monitoring dynamic integrity constraints based on temporal logic. *Information Systems*, 12(3):255–269.

Maskin, E. S. (2008). Mechanism design: How to implement social goals. *American Economic Review*, 98(3):567–576.

Mendelson, E. (1987). *Introduction to Mathematical Logic; (3rd Ed.)*. Wadsworth and Brooks/Cole Advanced Books & Software.

Meyer, J.-J. C. (1987). A different approach to deontic logic: deontic logic viewed as a variant of dynamic logic. *Notre Dame Journal of Formal Logic*, 29(1):109–136.

Meyer, J.-J. C. and Wieringa, R. (1994). *Deontic Logic in Computer Science: Normative System Specification*. Wiley.

Moore, J. (1992). Implementation, contracts and renegotiation in environments with complete information. *Advances in Economic Theory*, 1:182–282.

Moses, Y. and Tennenholtz, M. (1995). Artificial social systems. *Computers and AI*, 14(6):533–562.

Neumann, J. V. and Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Princeton University Press.

Nisan, N. (2007). *Algorithmic Game Theory*, chapter Introduction to Mechanism Design (for Computer Scientists), pages 209–242. Cambridge University Press.

Osborne, M. and Rubinstein, A. (1994). *A Course in Game Theory*. MIT Press.

Pauly, M. (2002). Programming and verifying subgame perfect mechanisms. *Journal of Logic and Computation*, 15.

Pnueli, A. (1977). The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57.

Prakken, H. and Sergot, M. J. (1996). Contrary-to-duty obligations. *Studia Logica*, 57(1):91–115.

Ramadge, P. J. and Wonham, W. M. (1987). Supervisory control of a class of discrete event processes. *SIAM J. Control Optim.*, 25(1):206–230.

Rao, A. S. and Georgeff, M. P. (1991). Modeling rational agents within a BDI architecture. In *2nd International Conference on Principles of Knowledge Representation and Reasoning*, KR '91, pages 473–484. Morgan Kaufmann Publishers.

Russell, S. J. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Pearson Education, 2nd edition.

Savitch, W. J. (1970). Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Science*, 4(2):177–192.

Schneider, K. (2004). *Verification of Reactive Systems: Formal Methods and Algorithms*. SpringerVerlag.

Searle, J. R. (1969). *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press.

Sergot, M. J. (2007). Action and agency in norm-governed multi-agent systems. In *Proceedings of the Engineering Societies in the Agents World VIII*, ESAW '07, pages 1–54.

Shoham, Y. (1993). Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92.

Shoham, Y. and Tennenholtz, M. (1992). 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.

Sistla, A. P. and Clarke, E. M. (1985). The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749.

Therborn, G. (2002). Back to norms! on the scope and dynamics of norms and normative action. *Current Sociology*, 50(6):863–880.

Tinnemeier, N. A. M., Dastani, M., Meyer, J.-J. C., and van der Torre, L. W. N. (2009). Programming normative artifacts with declarative obligations and prohibitions. In *IAT*, pages 145–152. IEEE.

van der Hoek, W., Jamroga, W., and Wooldridge, M. (2005). A logic for strategic reasoning. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '05, pages 157–164. ACM.

van der Hoek, W., Roberts, M., and Wooldridge, M. (2007). Social laws in alternating time: effectiveness, feasibility, and synthesis. *Synthese*, 156(1):1–19.

van Ditmarsch, H., van der Hoek, W., and Kooi, B. (2007). *Dynamic Epistemic Logic, volume 337 of Synthese Library Series*. Springer.

Verhagen, H. (2000). *Norm autonomous agents*. PhD thesis, Stockholm University.

von Wright, G. H. (1951). *An Essay in Modal Logic*. Amsterdam, North-Holland Pub. Co.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. Wiley Publishing, 2nd edition.

Woolridge, M. and van der Hoek, W. (2005). On obligations and normative ability. *Journal of Applied Logic*, 3:396–420.

Summary

A multi-agent system is a computerized system that is composed of multiple interacting agents within an environment. Such an agent can be a software program, but could equally well be a human actor. These systems are making their way into our everyday life, for example electronic marketplaces where agents can buy and sell goods, or smart grid energy systems where agents can produce or consume energy based on supply and demand. These systems are typically distributed and decentralized, and the participating agents may have diverging information, diverging interests, or both. The internals and architecture of these participating agents may be unknown to us, which disables us to make any strong assumption on the possible behaviour that these agents may exhibit. Moreover, these systems are generally composed or designed with a specific goal in mind. For example, an electronic marketplace is composed to facilitate safe transactions between currency and goods. Depending on the actual behaviour of the system these goals may, or may not, be achieved. Hence there is a need to regulate and control the behaviour of the system and the participating agents. In order to achieve this, norms have been proposed to regulate, coordinate, and control the behaviour of the agents and system, leading to the field of research called normative multi-agent systems. In this thesis we study enforcement norms, which are norms which can be violated, and which can lead to sanctions when a violation is detected. However, it can be the case that the norms are not well-designed, and that there exists ways for an agent to exploit the system. This may for example lead to dangerous situations within the system, or high costs for the designer. As these normative multi-agent systems can grow in size and scale, formal methods and tools are needed in order to prove whether such a system works as intended. This thesis aims to design and analyse formal logic-based frameworks in which normative multi-agent systems can be modelled and verified. Specifically, we design frameworks that aim to capture the behaviour of a normative multi-agent system, and we develop logics to reason about properties of such a system. Ultimately these logics can be used to verify the correctness of these systems.

As stated previously, the internal architecture of the participating agents

may be unknown to us, which implies that we cannot simply assume that the agents are aware of the norms, or compliant with respect to the norms. Moreover, agents may have preference over certain executions of the system, and these preferences may be unknown to us. However, once we make the assumption that the agents have some preference, and we are interested in how the agents behave, we enter the field of game theory. Game theory, in the broad sense, is the study of strategic decision making in the presence of (one or more) rational agent(s). Thus, a formal framework which allows us to model and verify normative multi-agent systems must be able to capture this intricate interplay between logic, norms and games that arises. This is the approach we have taken in Chapter 3 and Chapter 4. In Chapter 3 we developed a logic that allows reasoning of agents' strategic abilities under a multitude of compliance types. In this approach, the designer attributes compliance types to (coalitions of) agents, allowing us to verify whether the design objectives are satisfied in the system. In Chapter 4, we developed a logical proof system that allows us to specify and verify various implementability questions related to normative multi-agent systems. The idea is that by attributing types to agents, we were able to analyse the normative multi-agent system using game theoretic solution concepts. Since norms, once implemented, can change the environment of the system, these norms can act as a *mechanism* that can change the underlying game. Using this framework, we can study normative multi-agent systems using principles from *mechanism design*.

The approaches taken in Chapter 3 and Chapter 4 consider that the normative system is a static entity, that is, that the set of norms contained in the system are static and fixed throughout the execution of the system. However, in a typical normative system these norms may change over time, and these approaches do not cope with this fact. In this thesis we also want to model and verify normative multi-agent systems situated in a dynamic environment. In Chapter 5, we developed a dynamic logic and accompanying proof system to tackle this task.

Finally, in order to control the behaviour of the system and the participating agents, it is not only important to consider the norms, but also the monitors that are able to detect whenever a violation occurs. In other words, both the norms and the monitors serve as a control mechanism for the multi-agent system, and this thesis will focus on both. Monitors are important whenever the system as a whole is not fully observable. In Chapter 6, we developed such a framework in which monitors can be studied and verified. We provided a logical analysis of monitors, studied the relations between monitors and norms to be monitored, and explored computational aspects of verifying whether a monitor is sufficient for a normative multi-agent system.

Apart from the many theoretical results, an important contribution this thesis makes is that it bridges the gap between the field of *normative multi-*

agent systems and *the modelling and verification of formal systems*. From a practical perspective, this thesis sets a foundation for the development of tools in which normative multi-agent systems can be verified.

Samenvatting

Een multi-agent systeem is een gecomputeriseerd systeem dat uit meerdere interacterende agenten bestaat binnen een omgeving. Een dergelijke agent kan een softwareprogramma zijn, maar het zou bijvoorbeeld ook een menselijke speler kunnen zijn. Deze systemen worden steeds gangbaarder in ons alledaagse leven. Denk bijvoorbeeld aan een digitale marktplaats waar agenten goederen kunnen kopen en verkopen, of een *smart grid* energie systeem waar agenten energie kunnen produceren of consumeren, gebaseerd op vraag en aanbod. Dit soort systemen zijn vaak gedistribueerd en gedecentraliseerd, en de agenten in zulke systemen kunnen divergerende informatie, divergerende interesses, of beide bevatten. De interne architectuur van deze agenten is mogelijk onbekend, waardoor we niet altijd in staat zijn om te voorspellen wat voor gedrag een agent zal hebben. Bovendien zijn dit soort systemen vaak ontworpen met een specifiek doel in gedachten. Een digitale marktplaats is bijvoorbeeld ontworpen om een veilige transactie tussen geld en goederen te faciliteren. Afhankelijk van het daadwerkelijke gedrag van het systeem kan blijken dat deze doelen niet gewaarborgd zijn. Er bestaat dus een behoefte om het gedrag van het systeem en de agenten te reguleren en te coördineren. Om dit voor elkaar te krijgen kunnen normen gebruikt worden die specificeren wat de prohibities, obligaties en permissies van de agenten zijn. Wanneer een multi-agent systeem tevens een dergelijk normatief systeem bevat, spreken we van een normatief multi-agent systeem. In dit proefschrift beschouwen we handhavingsnormen, wat normen zijn die overtreden kunnen worden en die kunnen leiden tot sancties wanneer een overtreding gedetecteerd wordt. Echter, het kan het geval zijn dat de normen niet goed ontworpen zijn, en dat er manieren bestaan voor een agent om het systeem te exploiteren. Dit kan bijvoorbeeld leiden tot gevaarlijke situaties in het systeem, of hoge kosten voor de ontwerper. Naarmate deze systemen in grootte en schaal groeien zijn formele methoden nodig die aan kunnen tonen dat een systeem werkt zoals is bedoeld. Dit proefschrift houdt zich bezig met het ontwerpen en analyseren van formele logica-gebaseerde raamwerken waarin normatieve multi-agent systemen kunnen worden gemodelleerd en kunnen worden geverifieerd. In het bijzonder ontwerpen wij raamwerken die het gedrag van normatieve multi-

agent systemen vastleggen, en we ontwikkelen logica's die ons in staat stellen om eigenschappen te kunnen uitdrukken van het systeem. Uiteindelijk kunnen deze logica's gebruikt worden om de correctheid van een systeem te verifiëren.

Zoals eerder genoemd kan het zo zijn dat de interne architectuur van de agenten onbekend is, wat betekent dat we niet simpelweg kunnen aannemen dat de agenten zich bewust zijn van de normen, of de intentie hebben om zich te houden aan de normen. Bovendien zou het zo kunnen zijn dat agenten voorkeur hebben over bepaalde executies van het systeem en dat deze preferenties ook onbekend zijn. Echter, als we eenmaal de aanname maken dat agenten een bepaalde preferentie hebben en we geïnteresseerd zijn in hoe de agenten zich zullen gedragen, betreden we het onderzoeksveld van speltheorie. Speltheorie is een tak van wiskunde waarin de strategische interactie tussen agenten bestudeerd wordt. Een formeel raamwerk wat ons in staat stelt om normatieve multi-agent systemen te modelleren en verifiëren moet dus het samenspel tussen logica, normen en spellen kunnen omvatten. Dit is de aanpak die we hebben gehanteerd in Hoofdstuk 3 en Hoofdstuk 4. In Hoofdstuk 3 hebben we een logica ontwikkeld die het mogelijk maakt om te redeneren over de strategische capaciteiten van een agent onder aanname van verscheidende conformiteitstypes. Dit stelt een ontwerper in staat om de correctheid van een normatief multi-agent systeem te verifiëren onder aanname dat de agenten een bepaalde conformiteit hebben met betrekking tot de normen. In Hoofdstuk 4 hebben we een logica-gebaseerd bewijssysteem ontwikkeld wat het mogelijk maakt om verscheidende implementatiekwesties te specificeren en verifiëren. Het idee is dat we types associëren met agenten, wat ons in staat stelt om een normatief multi-agent systeem te analyseren met behulp van speltheoretische oplossingsconcepten. Sinds normen, wanneer ze geïmplementeerd worden, de omgeving van een systeem kunnen veranderen, kunnen normen gezien worden als een *mechanisme* die het onderliggende spel aanpast. Dit raamwerk maakt het mogelijk om normatieve multi-agent systemen te bestuderen in de context van *mechanisme ontwerp theorie*.

De aanpakken die we beschouwen in Hoofdstuk 3 en Hoofdstuk 4 nemen aan dat het normatieve systeem een statische entiteit is. Dat wil zeggen dat de set van normen bevat in het systeem statisch is gedurende de executie van het systeem. Echter, in een typisch alledaags normatief systeem zouden deze normen kunnen veranderen over de tijd, en de eerder genoemde benaderingen houden hier geen rekening mee. In dit proefschrift willen we ook normatieve multi-agent systemen modelleren en verifiëren die gesitueerd zijn in een dynamische omgeving. In Hoofdstuk 5 hebben we een dynamische logica en bijgaand bewijssysteem ontwikkeld om deze taak te volbrengen.

Ten slotte, om het gedrag van een systeem en de agenten te reguleren is het niet alleen belangrijk om de normen te beschouwen, maar ook de observatiemechanismen, oftewel *monitors*, die in staat zijn om te detecteren wanneer er

een overtreding plaats vindt. Met andere woorden, zowel de normen als de monitors dienen als een regulatiemechanisme voor het multi-agent systeem, en in dit proefschrift richten wij ons op beide. Monitors zijn belangrijk wanneer het systeem als geheel niet volledig geobserveerd kan worden. In Hoofdstuk 6 hebben wij een raamwerk ontwikkeld waarin monitors bestudeerd en geverifieerd kunnen worden. We geven een logische analyse van monitors, bestuderen de relaties tussen monitors en normen die gemonitord moeten worden, en verkennen computationele problemen omtrent het probleem om te verifiëren of een monitor geschikt is voor een normatief multi-agent systeem.

Afgezien van de vele theoretische resultaten is een belangrijke bijdrage die dit proefschrift maakt de overbrugging tussen het vakgebied van *normatieve multi-agent systemen* en *het modelleren en verifiëren van formele systemen*. Vanuit praktisch oogpunt wordt in dit proefschrift een basis gelegd voor de ontwikkeling van hulpmiddelen waarmee normatieve multi-agent systemen kunnen worden geverifieerd.

Curriculum Vitae

2015 - Current	Postdoctoral researcher at Delft University of Technology (Group: Interactive Intelligence) In my research I am applying artificial intelligence techniques to the domain of smart energy grids using agent-based technology. My goal is to investigate automated decision making mechanisms in smart energy grids, particularly focusing on negotiation approaches.
2011-2015	PhD researcher at Utrecht University (Group: Intelligent Systems) In my research I investigated formal methods to model and verify normative multi-agent systems. This PhD thesis is the result of this work.
2008- 2011	Master Artificial Intelligence at Utrecht University Track: Agent Technology (Cum Laude)
2005-2008	Bachelor Kunstmatige Intelligentie at Utrecht University
1999-2005	Pre-university education at CSG Het Streek, Ede

Assisted courses[1]

Inleiding Adaptieve Systemen EN: "Introduction to Adaptive Systems"	2008, 2009, 2010, 2011, 2012, 2013
Wiskunde voor AI EN: "Mathematics for AI"	2008, 2009
Imperatief programmeren in Java voor CKI EN: "Imperative programming in Java for AI"	2007, 2008
Logica voor AI EN: "Logic for AI"	2010, 2013
Intelligente systemen EN: "Intelligent systems"	2012
Kunstmatige intelligentie EN: "Artificial intelligence"	2011

[1] Left column contains the course name, right column contains the academic year(s) in which I assisted the course. All courses were given at Utrecht University.

SIKS Dissertation Series

1998

Johan van den Akker, DEGAS - An Active, Temporal Database of Autonomous Objects, CWI, 1998-1

Floris Wiesman, Information Retrieval by Graphically Browsing Meta-Information, UM, 1998-2

Ans Steuten, A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective, TUD, 1998-3

Dennis Breuker, Memory versus Search in Games, UM, 1998-4

E.W. Oskamp, Computerondersteuning bij Straftoemeting, RUL, 1998-5

1999

Mark Sloof, Physiology of Quality Change Modelling; Automated modelling of Quality Change of Agricultural Products, VU, 1999-1

Rob Potharst, Classification using decision trees and neural nets, EUR, 1999-2

Don Beal, The Nature of Minimax Search, UM, 1999-3

Jacques Penders, The practical Art of Moving Physical Objects, UM, 1999-4

Aldo de Moor, Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems, KUB, 1999-5

Niek J.E. Wijngaards, Re-design of compositional systems, VU, 1999-6

David Spelt, Verification support for object database design, UT, 1999-7

Jacques H.J. Lenting, Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation, UM, 1999-8

2000

Frank Niessink, Perspectives on Improving Software Maintenance, VU, 2000-1

Koen Holtman, Prototyping of CMS Storage Management, TUE, 2000-2

Carolien M.T. Metselaar, Sociaal-organisatorische gevolgen van kennistechnologie; een procesbenadering en actorperspectief, UVA, 2000-3

Geert de Haan, ETAG, A Formal Model of Competence Knowledge for User Interface Design, VU, 2000-4

Ruud van der Pol, Knowledge-based Query Formulation in Information Retrieval, UM, 2000-5

Rogier van Eijk, Programming Languages for Agent Communication, UU, 2000-6

Niels Peek, Decision-theoretic Planning of Clinical Patient Management, UU, 2000-7

Veerle Coup, Sensitivity Analyis of Decision-Theoretic Networks, EUR, 2000-8

Florian Waas, Principles of Probabilistic Query Optimization, CWI, 2000-9

Niels Nes, Image Database Management System Design Considerations, Algorithms and Architecture, CWI, 2000-10

Jonas Karlsson, Scalable Distributed Data Structures for Database Management, CWI, 2000-11

2001

Silja Renooij, Qualitative Approaches to Quantifying Probabilistic Networks, UU, 2001-1

Koen Hindriks, Agent Programming Languages: Programming with Mental Models, UU, 2001-2

Maarten van Someren, Learning as problem solving, UVA, 2001-3

Evgueni Smirnov, Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets, UM, 2001-4

Jacco van Ossenbruggen, Processing Structured Hypermedia: A Matter of Style, VU, 2001-5

Martijn van Welie, Task-based User Interface Design, VU, 2001-6

Bastiaan Schonhage, Diva: Architectural Perspectives on Information Visualization, VU, 2001-7

Pascal van Eck, A Compositional Semantic Structure for Multi-Agent Systems Dynamics, VU, 2001-8

Pieter Jan 't Hoen, Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes, RUL, 2001-9

Maarten Sierhuis, Modeling and Simulating Work Practice BRAHMS: a multiagent modeling and simulation language for work practice analysis and design, UVA, 2001-10

Tom M. van Engers, Knowledge Management: The Role of Mental Models in Business Systems Design, VUA, 2001-11

2002

Nico Lassing, Architecture-Level Modifiability Analysis, VU, 2002-01

Roelof van Zwol, Modelling and searching web-based document collections, UT, 2002-02

Henk Ernst Blok, Database Optimization Aspects for Information Retrieval, UT, 2002-03

Juan Roberto Castelo Valdueza, The Discrete Acyclic Digraph Markov Model in Data Mining, UU, 2002-04

Radu Serban, The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents, VU, 2002-05

Laurens Mommers, Applied legal epistemology; Building a knowledge-based ontology of the legal domain, UL, 2002-06

Peter Boncz, Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications, CWI, 2002-07

Jaap Gordijn, Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas, VU, 2002-08

Willem-Jan van den Heuvel, Integrating Modern Business Applications with Objectified Legacy Systems, KUB, 2002-09

Brian Sheppard, Towards Perfect Play of Scrabble, UM, 2002-10

Wouter C.A. Wijngaards, Agent Based Modelling of Dynamics: Biological and Organisational Applications, VU, 2002-11

Albrecht Schmidt, Processing XML in Database Systems, UVA, 2002-12

Hongjing Wu, A Reference Architecture for Adaptive Hypermedia Applications, TUE, 2002-13

Wieke de Vries, Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems, UU, 2002-14

Rik Eshuis, Semantics and Verification of UML Activity Diagrams for Workflow Modelling, UT, 2002-15

Pieter van Langen, The Anatomy of Design: Foundations, Models and Applications, VU, 2002-16

Stefan Manegold, Understanding, Modeling, and Improving Main-Memory Database Performance, UVA, 2002-17

2003

Heiner Stuckenschmidt, Onotology-Based Information Sharing In Weakly Structured Environments, VU, 2003-01

Jan Broersen, Modal Action Logics for Reasoning About Reactive Systems, VU, 2003-02

Martijn Schuemie, Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy, TUD, 2003-03

Milan Petkovic, Content-Based Video Retrieval Supported by Database Technology, UT, 2003-04

Jos Lehmann, Causation in Artificial Intelligence and Law - A modelling approach, UVA, 2003-05

Boris van Schooten, Development and specification of virtual environments, UT, 2003-06

Machiel Jansen, Formal Explorations of Knowledge Intensive Tasks, UVA, 2003-07

Yongping Ran, Repair Based Scheduling, UM, 2003-08

Rens Kortmann, The resolution of visually guided behaviour, UM, 2003-09

Andreas Lincke, Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture, UVT, 2003-10

Simon Keizer, Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks, UT, 2003-11

Roeland Ordelman, Dutch speech recognition in multimedia information retrieval, UT, 2003-12

Jeroen Donkers, Nosce Hostem - Searching with Opponent Models, UM, 2003-13

Stijn Hoppenbrouwers, Freezing Language: Conceptualisation Processes across ICT-Supported Organisations, KUN, 2003-14

Mathijs de Weerdt, Plan Merging in Multi-Agent Systems, TUD, 2003-15

Menzo Windhouwer, Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses, CWI, 2003-16

David Jansen, Extensions of Statecharts with Probability, Time, and Stochastic Timing, UT, 2003-17

Levente Kocsis, Learning Search Decisions, UM, 2003-18

2004

Virginia Dignum, A Model for Organizational Interaction: Based on Agents, Founded in Logic, UU, 2004-01

Lai Xu, Monitoring Multi-party Contracts for E-business, UVT, 2004-02

Perry Groot, A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving, VU, 2004-03

Chris van Aart, Organizational Principles for Multi-Agent Architectures, UVA, 2004-04

Viara Popova, Knowledge discovery and monotonicity, EUR, 2004-05

Bart-Jan Hommes, The Evaluation of Business Process Modeling Techniques, TUD, 2004-06

Elise Boltjes, Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes, UM, 2004-07

Joop Verbeek, Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politiële gegevensuitwisseling en digitale expertise, UM, 2004-08

Martin Caminada, For the Sake of the Argument; explorations into argument-based reasoning, VU, 2004-09

Suzanne Kabel, Knowledge-rich indexing of learning-objects, UVA, 2004-10

Michel Klein, Change Management for Distributed Ontologies, VU, 2004-11

The Duy Bui, Creating emotions and facial expressions for embodied agents, UT, 2004-12

Wojciech Jamroga, Using Multiple Models of Reality: On Agents who Know how to Play, UT, 2004-13

Paul Harrenstein, Logic in Conflict. Logical Explorations in Strategic Equilibrium, UU, 2004-14

Arno Knobbe, Multi-Relational Data Mining, UU, 2004-15

Federico Divina, Hybrid Genetic Relational Search for Inductive Learning, VU, 2004-16

Mark Winands, Informed Search in Complex Games, UM, 2004-17

Vania Bessa Machado, Supporting the Construction of Qualitative Knowledge Models, UVA, 2004-18

Thijs Westerveld, Using generative probabilistic models for multimedia retrieval, UT, 2004-19

Madelon Evers, Learning from Design: facilitating multidisciplinary design teams, Nyenrode, 2004-20

2005

Floor Verdenius, Methodological Aspects of Designing Induction-Based Applications, UVA, 2005-01

Erik van der Werf, AI techniques for the game of Go, UM, 2005-02

Franc Grootjen, A Pragmatic Approach to the Conceptualisation of Language, RUN, 2005-03

Nirvana Meratnia, Towards Database Support for Moving Object data, UT, 2005-04

Gabriel Infante-Lopez, Two-Level Probabilistic Grammars for Natural Language Parsing, UVA, 2005-05

Pieter Spronck, Adaptive Game AI, UM, 2005-06

Flavius Frasincar, Hypermedia Presentation Generation for Semantic Web Information Systems, TUE, 2005-07

Richard Vdovjak, A Model-driven Approach for Building Distributed Ontology-based Web Applications, TUE, 2005-08

Jeen Broekstra, Storage, Querying and Inferencing for Semantic Web Languages, VU, 2005-09

Anders Bouwer, Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments, UVA, 2005-10

Elth Ogston, Agent Based Matchmaking and Clustering - A Decentralized Approach to Search, VU, 2005-11

Csaba Boer, Distributed Simulation in Industry, EUR, 2005-12

Fred Hamburg, Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen, UL, 2005-13

Borys Omelayenko, Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics, VU, 2005-14

Tibor Bosse, Analysis of the Dynamics of Cognitive Processes, VU, 2005-15

Joris Graaumans, Usability of XML Query Languages, UU, 2005-16

Boris Shishkov, Software Specification Based on Re-usable Business Components, TUD, 2005-17

Danielle Sent, Test-selection strategies for probabilistic networks, UU, 2005-18

Michel van Dartel, Situated Representation, UM, 2005-19

Cristina Coteanu, Cyber Consumer Law, State of the Art and Perspectives, UL, 2005-20

Wijnand Derks, Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics, UT, 2005-21

2006

Samuil Angelov, Foundations of B2B Electronic Contracting, TUE, 2006-01

Cristina Chisalita, Contextual issues in the design and use of information technology in organizations, VU, 2006-02

Noor Christoph, The role of metacognitive skills in learning to solve problems, UVA, 2006-03

Marta Sabou, Building Web Service Ontologies, VU, 2006-04

Cees Pierik, Validation Techniques for Object-Oriented Proof Outlines, UU, 2006-05

Ziv Baida, Software-aided Service Bundling - Intelligent Methods & Tools for Graphical Service Modeling, VU, 2006-06

Marko Smiljanic, XML schema matching - balancing efficiency and effectiveness by means of clustering, UT, 2006-07

Eelco Herder, Forward, Back and Home Again - Analyzing User Behavior on the Web, UT, 2006-08

Mohamed Wahdan, Automatic Formulation of the Auditor's Opinion, UM, 2006-09

Ronny Siebes, Semantic Routing in Peer-to-Peer Systems, VU, 2006-10

Joeri van Ruth, Flattening Queries over Nested Data Types, UT, 2006-11

Bert Bongers, Interactivation - Towards an ecology of people, our technological environment, and the arts, VU, 2006-12

Henk-Jan Lebbink, Dialogue and Decision Games for Information Exchanging Agents, UU, 2006-13

Johan Hoorn, Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change, VU, 2006-14

Rainer Malik, CONAN: Text Mining in the Biomedical Domain, UU, 2006-15

Carsten Riggelsen, Approximation Methods for Efficient Learning of Bayesian Networks, UU, 2006-16

Stacey Nagata, User Assistance for Multitasking with Interruptions on a Mobile Device, UU, 2006-17

Valentin Zhizhkun, Graph transformation for Natural Language Processing, UVA, 2006-18

Birna van Riemsdijk, Cognitive Agent Programming: A Semantic Approach, UU, 2006-19

Marina Velikova, Monotone models for prediction in data mining, UVT, 2006-20

Bas van Gils, Aptness on the Web, RUN, 2006-21

Paul de Vrieze, Fundaments of Adaptive Personalisation, RUN, 2006-22

Ion Juvina, Development of Cognitive Model for Navigating on the Web, UU, 2006-23

Laura Hollink, Semantic Annotation for Retrieval of Visual Resources, VU, 2006-24

Madalina Drugan, Conditional log-likelihood MDL and Evolutionary MCMC, UU, 2006-25

Vojkan Mihajlovic, Score Region Algebra: A Flexible Framework for Structured Information Retrieval, UT, 2006-26

Stefano Bocconi, Vox Populi: generating video documentaries from semantically annotated media repositories, CWI, 2006-27

Borkur Sigurbjornsson, Focused Information Access using XML Element Retrieval, UVA, 2006-28

2007

Kees Leune, Access Control and Service-Oriented Architectures, UVT, 2007-01

Wouter Teepe, Reconciling Information Exchange and Confidentiality: A Formal Approach, RUG, 2007-02

Peter Mika, Social Networks and the Semantic Web, VU, 2007-03

Jurriaan van Diggelen, Achieving Semantic Interoperability in Multi-agent Systems: A Dialogue-based Approach, UU, 2007-04

Bart Schermer, Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance, UL, 2007-05

Gilad Mishne, Applied Text Analytics for Blogs, UVA, 2007-06

Natasa Jovanovic, To Who It May Concern - Addressee Identification in Face-to-Face Meetings, UT, 2007-07

Mark Hoogendoorn, Modeling of Change in Multi-Agent Organizations, VU, 2007-08

David Mobach, Agent-Based Mediated Service Negotiation, VU, 2007-09

Huib Aldewereld, Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols, UU, 2007-10

Natalia Stash, Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System, TUE, 2007-11

Marcel van Gerven, Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty, RUN, 2007-12

Rutger Rienks, Meetings in Smart Environments; Implications of Progressing Technology, UT, 2007-13

Niek Bergboer, Context-Based Image Analysis, UM, 2007-14

Joyca Lacroix, NIM: a Situated Computational Memory Model, UM, 2007-15

Davide Grossi, Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems, UU, 2007-16

Theodore Charitos, Reasoning with Dynamic Networks in Practice, UU, 2007-17

Bart Orriens, On the development an management of adaptive business collaborations, UVT, 2007-18

David Levy, Intimate relationships with artificial partners, UM, 2007-19

Slinger Jansen, Customer Configuration Updating in a Software Supply Network, UU, 2007-20

Karianne Vermaas, Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005, UU, 2007-21

Zlatko Zlatev, Goal-oriented design of value and process models from patterns, UT, 2007-22

Peter Barna, Specification of Application Logic in Web Information Systems, TUE, 2007-23

Georgina Ramrez Camps, Structural Features in XML Retrieval, CWI, 2007-24

Joost Schalken, Empirical Investigations in Software Process Improvement, VU, 2007-25

2008

Katalin Boer-Sorbn, Agent-Based Simulation of Financial Markets: A modular,continuous-time approach, EUR, 2008-01

Alexei Sharpanskykh, On Computer-Aided Methods for Modeling and Analysis of Organizations, VU, 2008-02

Vera Hollink, Optimizing hierarchical menus: a usage-based approach, UVA, 2008-03

Ander de Keijzer, Management of Uncertain Data - towards unattended integration, UT, 2008-04

Bela Mutschler, Modeling and simulating causal dependencies on process-aware information systems from a cost perspective, UT, 2008-05

Arjen Hommersom, On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective, RUN, 2008-06

Peter van Rosmalen, Supporting the tutor in the design and support of adaptive e-learning, OU, 2008-07

Janneke Bolt, Bayesian Networks: Aspects of Approximate Inference, UU, 2008-08

Christof van Nimwegen, The paradox of the guided user: assistance can be counter-effective, UU, 2008-09

Wauter Bosma, Discourse oriented summarization, UT, 2008-10

Vera Kartseva, Designing Controls for Network Organizations: A Value-Based Approach, VU, 2008-11

Jozsef Farkas, A Semiotically Oriented Cognitive Model of Knowledge Representation, RUN, 2008-12

Caterina Carraciolo, Topic Driven Access to Scientific Handbooks, UVA, 2008-13

Arthur van Bunningen, Context-Aware Querying; Better Answers with Less Effort, UT, 2008-14

Martijn van Otterlo, The Logic of Adaptive Behavior: Knowledge Representation and Algorithms for the Markov Decision Process Framework in First-Order Domains, UT, 2008-15

Henriette van Vugt, Embodied agents from a user's perspective, VU, 2008-16

Martin Op 't Land, Applying Architecture and Ontology to the Splitting and Allying of Enterprises, TUD, 2008-17

Guido de Croon, Adaptive Active Vision, UM, 2008-18

Henning Rode, From Document to Entity Retrieval: Improving Precision and Performance of Focused Text Search, UT, 2008-19

Rex Arendsen, Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met de overheid op de administratieve lasten van bedrijven, UVA, 2008-20

Krisztian Balog, People Search in the Enterprise, UVA, 2008-21

Henk Koning, Communication of IT-Architecture, UU, 2008-22

Stefan Visscher, Bayesian network models for the management of ventilator-associated pneumonia, UU, 2008-23

Zharko Aleksovski, Using background knowledge in ontology matching, VU, 2008-24

Geert Jonker, Efficient and Equitable Exchange in Air Traffic Management Plan Repair using Spender-signed Currency, UU, 2008-25

Marijn Huijbregts, Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled, UT, 2008-26

Hubert Vogten, Design and Implementation Strategies for IMS Learning Design, OU, 2008-27

Ildiko Flesch, On the Use of Independence Relations in Bayesian Networks, RUN, 2008-28

Dennis Reidsma, Annotations and Subjective Machines - Of Annotators, Embodied Agents, Users, and Other Humans, UT, 2008-29

Wouter van Atteveldt, Semantic Network Analysis: Techniques for Extracting, Representing and Querying Media Content, VU, 2008-30

Loes Braun, Pro-Active Medical Information Retrieval, UM, 2008-31

Trung H. Bui, Toward Affective Dialogue Management using Partially Observable Markov Decision Processes, UT, 2008-32

Frank Terpstra, Scientific Workflow Design; theoretical and practical issues, UVA, 2008-33

Jeroen de Knijf, Studies in Frequent Tree Mining, UU, 2008-34

Ben Torben Nielsen, Dendritic morphologies: function shapes structure, UVT, 2008-35

2009

Rasa Jurgelenaite, Symmetric Causal Independence Models, RUN, 2009-01

Willem Robert van Hage, Evaluating Ontology-Alignment Techniques, VU, 2009-02

Hans Stol, A Framework for Evidence-based Policy Making Using IT, UVT, 2009-03

Josephine Nabukenya, Improving the Quality of Organisational Policy Making using Collaboration Engineering, RUN, 2009-04

Sietse Overbeek, Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality, RUN, 2009-05

Muhammad Subianto, Understanding Classification, UU, 2009-06

Ronald Poppe, Discriminative Vision-Based Recovery and Recognition of Human Motion, UT, 2009-07

Volker Nannen, Evolutionary Agent-Based Policy Analysis in Dynamic Environments, VU, 2009-08

Benjamin Kanagwa, Design, Discovery and Construction of Service-oriented Systems, RUN, 2009-09

Jan Wielemaker, Logic programming for knowledge-intensive interactive applications, UVA, 2009-10

Alexander Boer, Legal Theory, Sources of Law & the Semantic Web, UVA, 2009-11

Peter Massuthe, Perating Guidelines for Services, TUE, Humboldt-Universitaet zu Berlin, 2009-12

Steven de Jong, Fairness in Multi-Agent Systems, UM, 2009-13

Maksym Korotkiy, From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA), VU, 2009-14

Rinke Hoekstra, Ontology Representation - Design Patterns and Ontologies that Make Sense, UVA, 2009-15

Fritz Reul, New Architectures in Computer Chess, UVT, 2009-16

Laurens van der Maaten, Feature Extraction from Visual Data, UVT, 2009-17

Fabian Groffen, Armada, An Evolving Database System, CWI, 2009-18

Valentin Robu, Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets, CWI, 2009-19

Bob van der Vecht, Adjustable Autonomy: Controling Influences on Decision Making, UU, 2009-20

Stijn Vanderlooy, Ranking and Reliable Classification, UM, 2009-21

Pavel Serdyukov, Search For Expertise: Going beyond direct evidence, UT, 2009-22

Peter Hofgesang, Modelling Web Usage in a Changing Environment, VU, 2009-23

Annerieke Heuvelink, Cognitive Models for Training Simulations, VUA, 2009-24

Alex van Ballegooij, "RAM: Array Database Management through Relational Mapping", CWI, 2009-25

Fernando Koch, An Agent-Based Model for the Development of Intelligent Mobile Services, UU, 2009-26

Christian Glahn, Contextual Support of Social Engagement and Reflection on the Web, OU, 2009-27

Sander Evers, Sensor Data Management with Probabilistic Models, UT, 2009-28

Stanislav Pokraev, Model-Driven Semantic Integration of Service-Oriented Applications, UT, 2009-29

Marcin Zukowski, Balancing vectorized query execution with bandwidth-optimized storage, CWI, 2009-30

Sofiya Katrenko, A Closer Look at Learning Relations from Text, UVA, 2009-31

Rik Farenhorst and Remco de Boer, Architectural Knowledge Management: Supporting Architects and Auditors, VU, 2009-32

Khiet Truong, How Does Real Affect Affect Affect Recognition In Speech?, UT, 2009-33

Inge van de Weerd, Advancing in Software Product Management: An Incremental Method Engineering Approach, UU, 2009-34

Wouter Koelewijn, Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling, UL, 2009-35

Marco Kalz, Placement Support for Learners in Learning Networks, OUN, 2009-36

Hendrik Drachsler, Navigation Support for Learners in Informal Learning Networks, OUN, 2009-37

Riina Vuorikari, Tags and self-organisation: a metadata ecology for learning resources in a multilingual context, OU, 2009-38

Christian Stahl, Service Substitution - A Behavioral Approach Based on Petri Nets, TUE, Humboldt-Universitaet zu Berlin, 2009-39

Stephan Raaijmakers, Multinomial Language Learning: Investigations into the Geometry of Language, UVT, 2009-40

Igor Berezhnyy, Digital Analysis of Paintings, UVT, 2009-41

Toine Bogers, Recommender Systems for Social Bookmarking, UVT, 2009-42

Virginia Nunes Leal Franqueira, Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients, UT, 2009-43

Roberto Santana Tapia, Assessing Business-IT Alignment in Networked Organizations, UT, 2009-44

Jilles Vreeken, Making Pattern Mining Useful, UU, 2009-45

Loredana Afanasiev, Querying XML: Benchmarks and Recursion, UVA, 2009-46

2010

Matthijs van Leeuwen, Patterns that Matter, UU, 2010-01

Ingo Wassink, Work flows in Life Science, UT, 2010-02

Joost Geurts, A Document Engineering Model and Processing Framework for Multimedia documents, CWI, 2010-03

Olga Kulyk, Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments, UT, 2010-04

Claudia Hauff, Predicting the Effectiveness of Queries and Retrieval Systems, UT, 2010-05

Sander Bakkes, Rapid Adaptation of Video Game AI, UVT, 2010-06

Wim Fikkert, A Gesture interaction at a Distance, UT, 2010-07

Krzysztof Siewicz, Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments, UL, 2010-08

Hugo Kielman, Politiële gegevensverwerking en Privacy, Naar een effectieve waarborging, UL, 2010-09

Rebecca Ong, Mobile Communication and Protection of Children, UL, 2010-10

Adriaan Ter Mors, The world according to MARP: Multi-Agent Route Planning, TUD, 2010-11

Susan van den Braak, Sensemaking software for crime analysis, UU, 2010-12

Gianluigi Folino, High Performance Data Mining using Bio-inspired techniques, RUN, 2010-13

Sander van Splunter, Automated Web Service Reconfiguration, VU, 2010-14

Lianne Bodenstaff, Managing Dependency Relations in Inter-Organizational Models, UT, 2010-15

Sicco Verwer, Efficient Identification of Timed Automata, theory and practice, TUD, 2010-16

Spyros Kotoulas, Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications, VU, 2010-17

Charlotte Gerritsen, Caught in the Act: Investigating Crime by Agent-Based Simulation, VU, 2010-18

Henriette Cramer, People's Responses to Autonomous and Adaptive Systems, UVA, 2010-19

Ivo Swartjes, Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative, UT, 2010-20

Harold van Heerde, Privacy-aware data management by means of data degradation, UT, 2010-21

Michiel Hildebrand, End-user Support for Access to Heterogeneous Linked Data, CWI, 2010-22

Bas Steunebrink, The Logical Structure of Emotions, UU, 2010-23

Dmytro Tykhonov, Designing Generic and Efficient Negotiation Strategies, TUD, 2010-24

Zulfiqar Ali Memon, Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective, VU, 2010-25

Ying Zhang, XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines, CWI, 2010-26

Marten Voulon, Automatisch contracteren, UL, 2010-27

Arne Koopman, Characteristic Relational Patterns, UU, 2010-28

Stratos Idreos, Database Cracking: Towards Auto-tuning Database Kernels, CWI, 2010-29

Marieke van Erp, Accessing Natural History - Discoveries in data cleaning, structuring, and retrieval, UVT, 2010-30

Victor de Boer, Ontology Enrichment from Heterogeneous Sources on the Web, UVA, 2010-31

Marcel Hiel, An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems, UVT, 2010-32

Robin Aly, Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval, UT, 2010-33

Teduh Dirgahayu, Interaction Design in Service Compositions, UT, 2010-34

Dolf Trieschnigg, Proof of Concept: Concept-based Biomedical Information Retrieval, UT, 2010-35

Jose Janssen, Paving the Way for Lifelong Learning; Facilitating competence development through a learning path specification, OU, 2010-36

Niels Lohmann, Correctness of services and their composition, TUE, 2010-37

Dirk Fahland, From Scenarios to components, TUE, 2010-38

Ghazanfar Farooq Siddiqui, Integrative modeling of emotions in virtual agents, VU, 2010-39

Mark van Assem, Converting and Integrating Vocabularies for the Semantic Web, VU, 2010-40

Guillaume Chaslot, Monte-Carlo Tree Search, UM, 2010-41

Sybren de Kinderen, Needs-driven service bundling in a multi-supplier setting - the computational e3-service approach, VU, 2010-42

Peter van Kranenburg, A Computational Approach to Content-Based Retrieval of Folk Song Melodies, UU, 2010-43

Pieter Bellekens, An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain, TUE, 2010-44

Vasilios Andrikopoulos, A theory and model for the evolution of software services, UVT, 2010-45

Vincent Pijpers, e3alignment: Exploring Inter-Organizational Business-ICT Alignment, VU, 2010-46

Chen Li, Mining Process Model Variants: Challenges, Techniques, Examples, UT, 2010-47

Withdrawn, 2010-48

Jahn-Takeshi Saito, Solving difficult game positions, UM, 2010-49

Bouke Huurnink, Search in Audiovisual Broadcast Archives, UVA, 2010-50

Alia Khairia Amin, Understanding and supporting information seeking tasks in multiple sources, CWI, 2010-51

Peter-Paul van Maanen, Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention, VU, 2010-52

Edgar Meij, Combining Concepts and Language Models for Information Access, UVA, 2010-53

2011

Botond Cseke, Variational Algorithms for Bayesian Inference in Latent Gaussian Models, RUN, 2011-01

Nick Tinnemeier, Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language, UU, 2011-02

Jan Martijn van der Werf, Compositional Design and Verification of Component-Based Information Systems, TUE, 2011-03

Hado van Hasselt, Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference learning algorithms, UU, 2011-04

Base van der Raadt, Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline, VU, 2011-05

Yiwen Wang, Semantically-Enhanced Recommendations in Cultural Heritage, TUE, 2011-06

Yujia Cao, Multimodal Information Presentation for High Load Human Computer Interaction, UT, 2011-07

Nieske Vergunst, BDI-based Generation of Robust Task-Oriented Dialogues, UU, 2011-08

Tim de Jong, Contextualised Mobile Media for Learning, OU, 2011-09

Bart Bogaert, Cloud Content Contention, UVT, 2011-10

Dhaval Vyas, Designing for Awareness: An Experience-focused HCI Perspective, UT, 2011-11

Carmen Bratosin, Grid Architecture for Distributed Process Mining, TUE, 2011-12

Xiaoyu Mao, Airport under Control. Multi-agent Scheduling for Airport Ground Handling, UVT, 2011-13

Milan Lovric, Behavioral Finance and Agent-Based Artificial Markets, EUR, 2011-14

Marijn Koolen, The Meaning of Structure: the Value of Link Evidence for Information Retrieval, UVA, 2011-15

Maarten Schadd, Selective Search in Games of Different Complexity, UM, 2011-16

Jiyin He, Exploring Topic Structure: Coherence, Diversity and Relatedness, UVA, 2011-17

Mark Ponsen, Strategic Decision-Making in complex games, UM, 2011-18

Ellen Rusman, The Mind ' s Eye on Personal Profiles, OU, 2011-19

Qing Gu, Guiding service-oriented software engineering - A view-based approach, VU, 2011-20

Linda Terlouw, Modularization and Specification of Service-Oriented Systems, TUD, 2011-21

Junte Zhang, System Evaluation of Archival Description and Access, UVA, 2011-22

Wouter Weerkamp, Finding People and their Utterances in Social Media, UVA, 2011-23

Herwin van Welbergen, Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior, UT, 2011-24

Syed Waqar ul Qounain Jaffry, Analysis and Validation of Models for Trust Dynamics, VU, 2011-25

Matthijs Aart Pontier, Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots, VU, 2011-26

Aniel Bhulai, Dynamic website optimization through autonomous management of design patterns, VU, 2011-27

Rianne Kaptein, Effective Focused Retrieval by Exploiting Query Context and Document Structure, UVA, 2011-28

Faisal Kamiran, Discrimination-aware Classification, TUE, 2011-29

Egon van den Broek, Affective Signal Processing (ASP): Unraveling the mystery of emotions, UT, 2011-30

Ludo Waltman, Computational and Game-Theoretic Approaches for Modeling Bounded Rationality, EUR, 2011-31

Nees-Jan van Eck, Methodological Advances in Bibliometric Mapping of Science, EUR, 2011-32

Tom van der Weide, Arguing to Motivate Decisions, UU, 2011-33

Paolo Turrini, Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations, UU, 2011-34

Maaike Harbers, Explaining Agent Behavior in Virtual Training, UU, 2011-35

Erik van der Spek, Experiments in serious game design: a cognitive approach, UU, 2011-36

Adriana Burlutiu, Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference, RUN, 2011-37

Nyree Lemmens, Bee-inspired Distributed Optimization, UM, 2011-38

Joost Westra, Organizing Adaptation using Agents in Serious Games, UU, 2011-39

Viktor Clerc, Architectural Knowledge Management in Global Software Development, VU, 2011-40

Luan Ibraimi, Cryptographically Enforced Distributed Data Access Control, UT, 2011-41

Michal Sindlar, Explaining Behavior through Mental State Attribution, UU, 2011-42

Henk van der Schuur, Process Improvement through Software Operation Knowledge, UU, 2011-43

Boris Reuderink, Robust Brain-Computer Interfaces, UT, 2011-44

Herman Stehouwer, Statistical Language Models for Alternative Sequence Selection, UVT, 2011-45

Beibei Hu, Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work, TUD, 2011-46

Azizi Bin Ab Aziz, Exploring Computational Models for Intelligent Support of Persons with Depression, VU, 2011-47

Mark Ter Maat, Response Selection and Turn-taking for a Sensitive Artificial Listening Agent, UT, 2011-48

Andreea Niculescu, Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality, UT, 2011-49

2012

Terry Kakeeto, Relationship Marketing for SMEs in Uganda, UVT, 2012-01

Muhammad Umair, Adaptivity, emotion, and Rationality in Human and Ambient Agent Models, VU, 2012-02

Adam Vanya, Supporting Architecture Evolution by Mining Software Repositories, VU, 2012-03

Jurriaan Souer, Development of Content Management System-based Web Applications, UU, 2012-04

Marijn Plomp, Maturing Interorganisational Information Systems, UU, 2012-05

Wolfgang Reinhardt, Awareness Support for Knowledge Workers in Research Networks, OU, 2012-06

Rianne van Lambalgen, When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions, VU, 2012-07

Gerben de Vries, Kernel Methods for Vessel Trajectories, UVA, 2012-08

Ricardo Neisse, Trust and Privacy Management Support for Context-Aware Service Platforms, UT, 2012-09

David Smits, Towards a Generic Distributed Adaptive Hypermedia Environment, TUE, 2012-10

J.C.B. Rantham Prabhakara, Process Mining in the Large: Preprocessing, Discovery, and Diagnostics, TUE, 2012-11

Kees van der Sluijs, Model Driven Design and Data Integration in Semantic Web Information Systems, TUE, 2012-12

Suleman Shahid, Fun and Face: Exploring non-verbal expressions of emotion during playful interactions, UVT, 2012-13

Evgeny Knutov, Generic Adaptation Framework for Unifying Adaptive Web-based Systems, TUE, 2012-14

Natalie van der Wal, Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes, VU, 2012-15

Fiemke Both, Helping people by understanding them - Ambient Agents supporting task execution and depression treatment, VU, 2012-16

Amal Elgammal, Towards a Comprehensive Framework for Business Process Compliance, UVT, 2012-17

Eltjo Poort, Improving Solution Architecting Practices, VU, 2012-18

Helen Schonenberg, What's Next? Operational Support for Business Process Execution, TUE, 2012-19

Ali Bahramisharif, Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing, RUN, 2012-20

Roberto Cornacchia, Querying Sparse Matrices for Information Retrieval, TUD, 2012-21

Thijs Vis, Intelligence, politie en veiligheidsdienst: verenigbare grootheden?, UVT, 2012-22

Christian Muehl, Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction, UT, 2012-23

Laurens van der Werff, Evaluation of Noisy Transcripts for Spoken Document Retrieval, UT, 2012-24

Silja Eckartz, Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application, UT, 2012-25

Emile de Maat, Making Sense of Legal Text, UVA, 2012-26

Hayrettin Gurkok, Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games, UT, 2012-27

Nancy Pascall, Engendering Technology Empowering Women, UVT, 2012-28

Almer Tigelaar, Peer-to-Peer Information Retrieval, UT, 2012-29

Alina Pommeranz, Designing Human-Centered Systems for Reflective Decision Making, TUD, 2012-30

Emily Bagarukayo, A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure, RUN, 2012-31

Wietske Visser, Qualitative multi-criteria preference representation and reasoning, TUD, 2012-32

Rory Sie, Coalitions in Cooperation Networks (COCOON), OUN, 2012-33

Pavol Jancura, Evolutionary analysis in PPI networks and applications, RUN, 2012-34

Evert Haasdijk, Never Too Old To Learn - Online Evolution of Controllers in Swarm- and Modular Robotics, VU, 2012-35

Denis Ssebugwawo, Analysis and Evaluation of Collaborative Modeling Processes, RUN, 2012-36

Agnes Nakakawa, A Collaboration Process for Enterprise Architecture Creation, RUN, 2012-37

Selmar Smit, Parameter Tuning and Scientific Testing in Evolutionary Algorithms, VU, 2012-38

Hassan Fatemi, Risk-aware design of value and coordination networks, UT, 2012-39

Agus Gunawan, Information Access for SMEs in Indonesia, UVT, 2012-40

Sebastian Kelle, Game Design Patterns for Learning, OU, 2012-41

Dominique Verpoorten, Reflection Amplifiers in self-regulated Learning, OU, 2012-42

Withdrawn, 2012-43

Anna Tordai, On Combining Alignment Techniques, VU, 2012-44

Benedikt Kratz, A Model and Language for Business-aware Transactions, UVT, 2012-45

Simon Carter, Exploration and Exploitation of Multilingual Data for Statistical Machine Translation, UVA, 2012-46

Manos Tsagkias, Mining Social Media: Tracking Content and Predicting Behavior, UVA, 2012-47

Jorn Bakker, Handling Abrupt Changes in Evolving Time-series Data, TUE, 2012-48

Michael Kaisers, Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions, UM, 2012-49

Steven van Kervel, Ontologogy driven Enterprise Information Systems Engineering, TUD, 2012-50

Jeroen de Jong, Heuristics in Dynamic Sceduling; a practical framework with a case study in elevator dispatching, TUD, 2012-51

2013

Viorel Milea, News Analytics for Financial Decision Support, EUR, 2013-01

Erietta Liarou, MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing, CWI, 2013-02

Szymon Klarman, Reasoning with Contexts in Description Logics, VU, 2013-03

Chetan Yadati, Coordinating autonomous planning and scheduling, TUD, 2013-04

Dulce Pumareja, Groupware Requirements Evolutions Patterns, UT, 2013-05

Romulo Goncalves, The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience, CWI, 2013-06

Giel van Lankveld, Quantifying Individual Player Differences, UVT, 2013-07

Robbert-Jan Merk, Making enemies: cognitive modeling for opponent agents in fighter pilot simulators, VU, 2013-08

Fabio Gori, Metagenomic Data Analysis: Computational Methods and Applications, RUN, 2013-09

Jeewanie Jayasinghe Arachchige, A Unified Modeling Framework for Service Design, UVT, 2013-10

Evangelos Pournaras, Multi-level Reconfigurable Self-organization in Overlay Services, TUD, 2013-11

Marian Razavian, Knowledge-driven Migration to Services, VU, 2013-12

Mohammad Safiri, Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly, UT, 2013-13

Jafar Tanha, Ensemble Approaches to Semi-Supervised Learning Learning, UVA, 2013-14

Daniel Hennes, Multiagent Learning - Dynamic Games and Applications, UM, 2013-15

Eric Kok, Exploring the practical benefits of argumentation in multi-agent deliberation, UU, 2013-16

Koen Kok, The PowerMatcher: Smart Coordination for the Smart Electricity Grid, VU, 2013-17

Jeroen Janssens, Outlier Selection and One-Class Classification, UVT, 2013-18

Renze Steenhuizen, Coordinated Multi-Agent Planning and Scheduling, TUD, 2013-19

Katja Hofmann, Fast and Reliable Online Learning to Rank for Information Retrieval, UVA, 2013-20

Sander Wubben, Text-to-text generation by monolingual machine translation, UVT, 2013-21

Tom Claassen, Causal Discovery and Logic, RUN, 2013-22

Patricio de Alencar Silva, Value Activity Monitoring, UVT, 2013-23

Haitham Bou Ammar, Automated Transfer in Reinforcement Learning, UM, 2013-24

Agnieszka Anna Latoszek-Berendsen, Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System, UM, 2013-25

Alireza Zarghami, Architectural Support for Dynamic Homecare Service Provisioning, UT, 2013-26

Mohammad Huq, Inference-based Framework Managing Data Provenance, UT, 2013-27

Frans van der Sluis, When Complexity becomes Interesting: An Inquiry into the Information eXperience, UT, 2013-28

Iwan de Kok, Listening Heads, UT, 2013-29

Joyce Nakatumba, Resource-Aware Business Process Management: Analysis and Support, TUE, 2013-30

Dinh Khoa Nguyen, Blueprint Model and Language for Engineering Cloud Applications, UVT, 2013-31

Kamakshi Rajagopal, Networking For Learning; The role of Networking in a Lifelong Learner's Professional Development, OUN, 2013-32

Qi Gao, User Modeling and Personalization in the Microblogging Sphere, TUD, 2013-33

Kien Tjin-Kam-Jet, Distributed Deep Web Search, UT, 2013-34

Abdallah El Ali, Minimal Mobile Human Computer Interaction, UVA, 2013-35

Than Lam Hoang, Pattern Mining in Data Streams, TUe, 2013-36

Dirk Brner, Ambient Learning Displays, OUN, 2013-37

Eelco den Heijer, Autonomous Evolutionary Art, VU, 2013-38

Joop de Jong, A Method for Enterprise Ontology based Design of Enterprise Information Systems, TUD, 2013-39

Pim Nijssen, Monte-Carlo Tree Search for Multi-Player Games, UM, 2013-40

Jochem Liem, Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning, UVA, 2013-41

Léon Planken, Algorithms for Simple Temporal Reasoning, TUD, 2013-42

Marc Bron, Exploration and Contextualization through Interaction and Concepts, UVA, 2013-43

2014

Nicola Barile, Studies in Learning Monotone Models from Data, UU, 2014-01

Fiona Tuliyano, Combining System Dynamics with a Domain Modeling Method, RUN, 2014-02

Sergio Raul Duarte Torres, Information Retrieval for Children: Search Behavior and Solutions, UT, 2014-03

Hanna Jochmann-Mannak, Websites for children: search strategies and interface design - Three studies on children's search performance and evaluation, UT, 2014-04

Jurriaan van Reijsen, Knowledge Perspectives on Advancing Dynamic Capability, UU, 2014-05

Damian Tamburri, Supporting Networked Software Development, VU, 2014-06

Arya Adriansyah, Aligning Observed and Modeled Behavior, TUE, 2014-07

Samur Araujo, Data Integration over Distributed and Heterogeneous Data Endpoints, TUD, 2014-08

Philip Jackson, Toward Human-Level Artificial Intelligence: Representation and Computation of Meaning in Natural Language, UVT, 2014-09

Ivan Salvador Razo Zapata, Service Value Networks, VU, 2014-10

Janneke van der Zwaan, An Empathic Virtual Buddy for Social Support, TUD, 2014-11

Willem van Willigen, Look Ma, No Hands: Aspects of Autonomous Vehicle Control, VU, 2014-12

Arlette van Wissen, Agent-Based Support for Behavior Change: Models and Applications in Health and Safety Domains, VU, 2014-13

Yangyang Shi, Language Models With Meta-information, TUD, 2014-14

Natalya Mogles, Agent-Based Analysis and Support of Human Functioning in Complex Socio-Technical Systems: Applications in Safety and Healthcare, VU, 2014-15

Krystyna Milian, Supporting trial recruitment and design by automatically interpreting eligibility criteria, VU, 2014-16

Kathrin Dentler, Computing healthcare quality indicators automatically: Secondary Use of Patient Data and Semantic Interoperability, VU, 2014-17

Mattijs Ghijsen, Methods and Models for the Design and Study of Dynamic Agent Organizations, VU, 2014-18

Vinicius Ramos, Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support, TUE, 2014-19

Mena Habib, Named Entity Extraction and Disambiguation for Informal Text: The Missing Link, UT, 2014-20

Kassidy Clark, Negotiation and Monitoring in Open Environments, TUD, 2014-21

Marieke Peeters, Personalized Educational Games - Developing agent-supported scenario-based training, UU, 2014-22

Eleftherios Sidirourgos, Space Efficient Indexes for the Big Data Era, UVA/CWI, 2014-23

Davide Ceolin, Trusting Semi-structured Web Data, VU, 2014-24

Martijn Lappenschaar, New network models for the analysis of disease interaction, RUN, 2014-25

Tim Baarslag, What to Bid and When to Stop, TUD, 2014-26

Rui Jorge Almeida, Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty, EUR, 2014-27

Anna Chmielowiec, Decentralized k-Clique Matching, VU, 2014-28

Jaap Kabbedijk, Variability in Multi-Tenant Enterprise Software, UU, 2014-29

Peter de Cock, Anticipating Criminal Behaviour, UVT, 2014-30

Leo van Moergestel, Agent Technology in Agile Multiparallel Manufacturing and Product Support, UU, 2014-31

Naser Ayat, On Entity Resolution in Probabilistic Data, UVA, 2014-32

Tesfa Tegegne, Service Discovery in eHealth, RUN, 2014-33

Christina Manteli, The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems, VU, 2014-34

Joost van Ooijen, Cognitive Agents in Virtual Worlds: A Middleware Design Approach, UU, 2014-35

Joos Buijs, Flexible Evolutionary Algorithms for Mining Structured Process Models, TUE, 2014-36

Maral Dadvar, Experts and Machines United Against Cyberbullying, UT, 2014-37

Danny Plass-Oude Bos, Making brain-computer interfaces better: improving usability through post-processing, UT, 2014-38

Jasmina Maric, Web Communities, Immigration, and Social Capital, UVT, 2014-39

Walter Omona, A Framework for Knowledge Management Using ICT in Higher Education, RUN, 2014-40

Frederic Hogenboom, Automated Detection of Financial Events in News Text, EUR, 2014-41

Carsten Eijckhof, Contextual Multidimensional Relevance Models, CWI/TUD, 2014-42

Kevin Vlaanderen, Supporting Process Improvement using Method Increments, UU, 2014-43

Paulien Meesters, Intelligent Blauw. Intelligence-gestuurde politiezorg in gebiedsgebonden eenheden, UVT, 2014-44

Birgit Schmitz, Mobile Games for Learning: A Pattern-Based Approach, OUN, 2014-45

Ke Tao, Social Web Data Analytics: Relevance, Redundancy, Diversity, TUD, 2014-46

Shangsong Liang, Fusion and Diversification in Information Retrieval, UVA, 2014-47

2015

Niels Netten, Machine Learning for Relevance of Information in Crisis Response, UVA, 2015-01

Faiza Bukhsh, Smart auditing: Innovative Compliance Checking in Customs Controls, UVT, 2015-02

Twan van Laarhoven, Machine learning for network data, RUN, 2015-03

Howard Spoelstra, Collaborations in Open Learning Environments, OUN, 2015-04

Christoph Bösch, Cryptographically Enforced Search Pattern Hiding, UT, 2015-05

Farideh Heidari, Business Process Quality Computation - Computing Non-Functional Requirements to Improve Business Processes, TUD, 2015-06

Maria-Hendrike Peetz, Time-Aware Online Reputation Analysis, UVA, 2015-07

Jie Jiang, Organizational Compliance: An agent-based model for designing and evaluating organizational interactions, TUD, 2015-08

Randy Klaassen, HCI Perspectives on Behavior Change Support Systems, UT, 2015-09

Henry Hermans, OpenU: design of an integrated system to support lifelong learning, OUN, 2015-10

Yongming Luo, Designing algorithms for big graph datasets: A study of computing bisimulation and joins, TUE, 2015-11

Julie M. Birkholz, Modi Operandi of Social Network Dynamics: The Effect of Context on Scientific Collaboration Networks, VU, 2015-12

Giuseppe Procaccianti, Energy-Efficient Software, VU, 2015-13

Bart van Straalen, A cognitive approach to modeling bad news conversations, UT, 2015-14

Klaas Andries de Graaf, Ontology-based Software Architecture Documentation, VU, 2015-15

Changyun Wei, Cognitive Coordination for Cooperative Multi-Robot Teamwork, UT, 2015-16

André van Cleeff, Physical and Digital Security Mechanisms: Properties, Combinations and Trade-offs, UT, 2015-17

Holger Pirk, Waste Not, Want Not! - Managing Relational Data in Asymmetric Memories, CWI, 2015-18

Bernardo Tabuenca, Ubiquitous Technology for Lifelong Learners, OUN, 2015-19

Loïs Vanhée, Using Culture and Values to Support Flexible Coordination, UU, 2015-20

Sibren Fetter, Using Peer-Support to Expand and Stabilize Online Learning, OUN, 2015-21

Zhemin Zhu, Co-occurrence Rate Networks, UT, 2015-22

Luit Gazendam, Cataloguer Support in Cultural Heritage, VU, 2015-23

Richard Berendsen, Finding People, Papers, and Posts: Vertical Search Algorithms and Evaluation, UVA, 2015-24

Steven Woudenberg, Bayesian Tools for Early Disease Detection, UU, 2015-25

Alexander Hogenboom, Sentiment Analysis of Text Guided by Semantics and Structure, EUR, 2015-26

Sándor Héman, Updating compressed column-stores, CWI, 2015-27

Janet Bagorogoza, Knowledge Management and High Performance; The Uganda Financial Institutions Model for HPO, TiU, 2015-28

Hendrik Baier, Monte-Carlo Tree Search Enhancements for One-Player and Two-Player Domains, UM, 2015-29

Kiavash Bahreini, Real-time Multimodal Emotion Recognition in E-Learning, OUN, 2015-30

Yakup Koç, On Robustness of Power Grids, TUD, 2015-31

Jerome Gard, Corporate Venture Management in SMEs, UL, 2015-32

Frederik Schadd, Ontology Mapping with Auxiliary Resources, UM, 2015-33

Victor de Graaff, Geosocial Recommender Systems, UT, 2015-34

Junchao Xu, Affective Body Language of Humanoid Robots: Perception and Effects in Human Robot Interaction, TUD, 2015-35

2016

Syed Saiden Abbas, Recognition of Shapes by Humans and Machines, RUN, 2016-01

Michiel Christiaan Meulendijk, Optimizing medication reviews through decision support: prescribing a better pill to swallow, UU, 2016-02

Maya Sappelli, Knowledge Work in Context: User Centered Knowledge Worker Support, RUN, 2016-03

Laurens Rietveld, Publishing and Consuming Linked Data, VU, 2016-04

Evgeny Sherkhonov, Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers, UVA, 2016-05

Michel Wilson, Robust scheduling in an uncertain environment, TUD, 2016-06

Jeroen de Man, Measuring and modeling negative emotions for virtual training, VU, 2016-07

Matje van de Camp, A Link to the Past: Constructing Historical Social Networks from Unstructured Data, TiU, 2016-08

Archana Nottamkandath, Trusting Crowdsourced Information on Cultural Artefacts, VU, 2016-09

George Karafotias, Parameter Control for Evolutionary Algorithms, VUA, 2016-10

Anne Schuth, Search Engines that Learn from Their Users, UVA, 2016-11

Max Knobbout, Logics for Modelling and Verifying Normative Multi-Agent Systems, UU, 2016-12