Autonomy vs. Conformity

an Institutional Perspective on
Norms and Protocols

Nederlandse Organisatie voor Wetenschappelijk Onderzoek
This research was supported by the Netherlands Organisation for Scientific Research (NWO) under project number 634.000.017

SIKS Dissertation Series No. 2007-10

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

ISBN 978-90-393-4546-7

Autonomy vs. Conformity

an Institutional Perspective on
Norms and Protocols

Autonomie tegenover Overeenstemming

Een institutioneel perspectief op
normen en protocollen

(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. W.H. Gispen,
ingevolge het besluit van het college voor promoties
in het openbaar te verdedigen
op maandag 4 juni 2007 des ochtends te 10.30 uur

door

Hubertus Maria Aldewereld

geboren op 18 februari 1980, te Heerlen

Promotor:	Prof. dr. J.-J.Ch. Meyer
Co-promotor:	Dr. F.P.M. Dignum

Dit proefschrift werd (mede) mogelijk gemaakt met financiele steun van de Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO).

Scientia gratia scientiae

Acknowledgements

About one and a half years ago when I started thinking about how to write my thesis and to present my research of the previous years, I decided that I would not write a lengthy Acknowledgement, containing all the well-know clichés that are always used. No “it has been hard, but now it is done” or “it has been a long trip with many bumps” or any other of those well-known, much used phrases. Everyone I talked to before I started my PhD already said that it was hard work, and that it would most likely involve many late nights and weekends spend on the research. At that time I also decided that the acknowledgements could be done quick and short; a simple “thanks to all for being there” would include everyone I ever wanted to thank, without the running the risk of forgetting someone. It would save me time on writing it, it saves the readers time, and it covers all I ever needed to include, win-win in my opinion at that time. Over the course of last year, during the writing of this thesis, I changed my mind on this, and realised that giving thanks is more about commending the support of people, not by just writing the words, but by realising how important all those people have been for the process. Although this is hard to show, the least I can do is spend some time on writing a proper acknowledgement.

The subject of the research has always been broad, agents are used in various different situations (leading to a broad selection of research) and norms have been studied by many different research areas already (see chapter 2). This aspect of the research has always appealed to me, as it allows one to expand his horizon and become familiar with theories (and workings) of other research areas. It was even more pleasurable to realise that the subject of the research is, in some points, related to works of literature that have inspired me for a long time.

The idea of using norms to regulate agents is a very old one. Already in the 1950s, in the works of the famous science-fiction writer Isaac Asimov, norms are introduced to control robots. Asimov envisioned that when robots became autonomous (in his time robots were a mere vision itself, and the few mechanical horrors that existed at that time had few or no form intelligence at all, not to mention any form of autonomy) some form of regulation is required to make the interaction between robots and humans more pleasant. In fact, the three laws that Asimov proposed in his books are there for the exact reason which has been pictured so many times in Hollywood movies about autonomous artificial intelligence and robots; to control the robots and make sure they do not harm humans or try to overthrow and enslave (or kill) humanity. Even today Asimov's three basic laws are often cited in talks about robotics and artificial intelligence as they lay the foundation of the moral code for artificial intelligence. The most striking aspect of Asimov's stories is the fact that the three laws, which are absolute and cannot be violated, lead to bad situations, which they would seemingly have prevented. Therefore a new kind of robot is introduced that has the capabilities to 'turn off' it's compliance to the three laws, allowing it to better handle situations where (extreme) conformity to the laws leads to bad results.

The balance between compliance and autonomy, pertaining to the fact that, although laws and regulations are required to control the interactions of agents (or robots), also reminds us of a major debate happening today i real life. With the ever increasing threat of terrorism, citizens of major countries have (been forced) to give up more and more of their privacy because the government needs (or thinks it needs) better and more forms of enforcement to prevent terrorist attacks. Of course, if this trend is driven to it's extreme, situations such as those presented in '1984' by George Orwell can occur where the government controls every bit of the life of the citizens. Likewise, if the balance between compliance and autonomy is driven to it's extreme, unwanted situations happen; resulting in an anarchy in case of too much autonomy, or resulting in a sort of communist state in case of too much conformity. The essence of the balance between conformity and autonomy (and likewise between control and privacy) is that increasing the former decreases the latter, and vice versa. If either is given too much focus (going into too much detail), the other diminishes and eventually disappears. This is precisely what can be seen in the picture by M.C. Escher on the cover. The birds and the fishes form a harmony together, but whenever either the birds (at the top) or the fishes (at the bottom) is coloured with all details, the other one has disappeared.

Over this last year while writing this thesis, and during the several years of research prior to that, many people have helped or supported me in different manners and I would like to thank them for it. Major thanks go to my family for all their support and accepting that I was not there for them when they were there for me. I want to mention especially my parents; thank you for everything, a good solid education, and although I may not show it all the time, you did a good job. I can only hope that you are proud of me. Thanks to my brothers as well, I know I have not often shown my appreciation, but it does not mean that I do not.

I would like to say special thanks to my promotor and co-promotor for taking the time and effort to guide me during the process of writing this thesis and it's accompanied research. Next to that I would like to thank NWO for the means to perform this research, and to my colleagues from the universities of Groningen, Maastricht and Leiden, who accompanied me in the ANITA project, thank you for a pleasant cooperation; maybe it has not been as fruitful as we had hoped, I did learn a lot from working together with you.

Of course, I would love to thank many other people that helped me in some way over the past few years. Thanks to my colleagues; the interesting discussion during the lunch breaks kept me going for quite some time, I will not forget those. Special thanks to my roommates, I have had quite a few come and go those four years; thanks for a welcome face every morning, and helping me with whatever questions I had (be it concerning logics or LaTeX, or something none work related, there was always time for it). Thanks to the colleagues at my new job for making me welcome and accepting me into your community. Also thanks to my dear friends; we know each other quite some time already and our discussions about philosophical and political issues, or other miscellaneous matters (photography, movies, games) have been a pleasant distraction from the research and hard work needed for completing this thesis. I am also glad that you provided me with the means to air my thoughts (and doubts) about the progress of my thesis. Thanks to my 'colleagues' from Aikido (although I consider you friends as well, you still deserve to be mentioned separately) for being patient enough with me all those times that I was working when I should have been training. Even though (especially near the end) I hardly ever came, I was always welcome, and you treated me as if I had been visiting as regularly as ever; I really appreciate it.

As always, there are more people that helped in one way or another that I did not mention. This does not make their support less appreciated or less important than those mentioned, and in some ways just being can mean as much as helping out on important issues that are research related.

Thanks to all.

Huib Aldewereld
April 2007

Contents

Chapter 1

Introduction

In recent years, multiagent systems have increased in number and are used for more complex domains every day. Serious problems arise, however, when agents are used in highly-regulated domains, which are domains that are governed by lots of complex and abstract laws. Agent solutions used for medical and judicial applications, for instance donor organ assignments or criminal information exchange, have to cope with the vast amount of laws and regulations that govern such domains. The laws of these domains, made to control and regulate the human interactions in the domain, naturally, also apply to any agent system that is created to automate those human interactions. Here two questions emerge. 1) How can we make sure that laws and regulations on the activities and information exchange in these domains are met? 2) How do we create mechanisms to enforce norm compliance and therefore increase trust between individuals and organisations?

In this thesis, we will try to answer those two questions. We propose a methodology for the design and implementation of electronic institutions based on human laws and regulations, and provide the formal links between 1) the norms expressed in law and 2) the norm enforcement and procedures used in the practice. Moreover, in order to ensure the norm compliance of the protocols used in agent institutions, we introduce a formal framework for the verification of the norm compliance of protocols.

This chapter is organised as follows. Section 1.1 provides the motivation for the research presented in this thesis. In section 1.2 the research approach is presented, along with the research questions and objectives. Finally, section 1.3 provides an overview of the following chapters.

1.1 Motivation

The motivation for the research presented in this thesis is the ANITA project, which is a multidisciplinary joint venture of legal and artificial intelligence departments of four universities in the Netherlands (consisting of Rijksuniversiteit Groningen, Universiteit Leiden, Universiteit Maastricht and Universiteit Utrecht). ANITA stands for Administrative Normative Information Transaction Agents, and aims to provide a solution, based on multiagent technology, for the problem that exists in information exchange between the different police districts in the Netherlands. In this domain the main challenges concern both the shortage of information (not being able to find legally relevant data that should be available) as well as abundance of information (for example violating privacy rights). These issues are of great consequence in the domain of police and judicial intelligence. The ANITA framework is based on administrative agents that decide, based on norms, whether to allow transactions of information.

The ANITA project is part of the NWO research programme ToKeN ('Toegankelijkheid en Kennisontsluiting Nederland', Access To Knowledge and its enhancement Netherlands), which involves specialists in both the fields of cognitive and computer sciences. The ToKeN programme focuses on the ability of individuals to retrieve relevant knowledge and information from computer systems and to derive implicit knowledge from raw data. The overall aim is solving the fundamental problems of the interaction between human users and knowledge and information systems.

In the ANITA scenario, where agents autonomously decide whether or not to share (privacy-sensitive) information based on the applicability of (local) norms, a global frame for the enforcement of (global) norms was needed. Since all agents, though each is bound to local procedures and rules, have to adhere to the global regulations as given by law (e.g., the police file act, and privacy laws), it has to be checked (at real-time) whether the information transactions are not in conflict with the global laws that hold for the domain. In most software and agent methodologies, such regulations are seen only as additional requirements in the analysis phase of the system, and are thus hard-coded into the software or the agents themselves. If, however, the regulations change, it becomes hard to track all changes to be done in the implementation, as there is no explicit representation of the norms (that is, since all norms are embedded in the agent's design and code, all the design steps have to be checked and all the code verified to ensure compliance to the new regulations). The alternative is to have an explicit representation of the norms, but this approach requires some form of enforcement to ensure the compliance instead.

Research on intelligent agents has created the concept of electronic institutions (e-institutions for short). As their human counterparts, an electronic institution is an entity defining a set of norms over the behaviour of individuals inside the institution. Recent research in electronic institutions has focused on the use of software agent technology, to create the concept of *agent-mediated electronic institutions.*

E-institutions belong to a new and promising field where interactions between groups of agents are regulated by means of a set of explicit norms, expressed in a computational language that agents can interpret. An e-institution [Dignum and Dignum, 2001; Esteva et al., 2001] is a safe environment mediating the interaction of agents. The expected behaviour of agents in such an environment is described by means of an explicit specification of norms, which is expressive enough for its purpose, readable by agents, and easy to maintain.

Using an e-institution in a highly-regulated scenario such as the ANITA scenario requires us to solve the following four issues.

1. **Abstractness of human regulations**: human regulations are written in an abstract way, open to several interpretations to make the legal text stable for a long time. The abstraction poses a problem when trying to implement the regulations on computers, where meanings should be precise and unambiguous.

2. **Operationalisation of norms**: usually norms are formally specified in languages based in deontic logic, which is expressive enough to cover some of the concepts that appear in regulations (obligations, permissions, prohibitions), has formal semantics and allows the verification of sets of norms. However, deontic-based languages are declarative in nature and do not fully express the operational impact of norms on the behaviour of the multiagent system.

3. **Implementation of norms from the institutional perspective**: agent platforms should be extended with mechanisms to detect attempts from agents to break the norms. This issue is not only relevant in open scenarios where non-trusted agents may enter the system, but also in scenarios with trusted agents, as sometimes these agents may break the norms unwillingly (e.g., not detecting that a norm applied in a given situation).

4. **Methodology to design electronic institutions**: although there are currently some toolkits and frameworks to build electronic institutions, there is no methodology that covers all aspects, from the specification of the abstract norms to the connection with their implementation, both from the institutional and agent viewpoints.

An additional issue would be to have agents which can understand a given set of norms and are able to reason about the compliance of their behaviour against the norms. Although it would be desirable to have such agents it is not crucial for the success of e-institutions, as the implementation of the norms from an institutional perspective (issue 3) ensures the compliance of the agents, and we cannot assume in an open scenario that all agents interacting with an e-institution would be capable of normative reasoning to ensure the proper behaviour themselves.

1.2 Research Approach

In this thesis, we investigate the implementation of norms in open multiagent systems. More specifically, we are interested in the design and implementation of electronic institutions based on human laws and regulations. In pursuit of this goal we investigate how concepts in norms can be translated to concrete concepts used in practice (issue 1), how the norms can be implemented by giving them an operational meaning (issue 2), and how norms can be used to design protocols that can be used by agents in the institution (issue 3).

Due to the nature of open environments where agents with different goals and architectures co-exist, no guarantees can be given about the inner workings of the agents. Our main assumption is that none of the agents are necessarily able to reason about norms and an enforcement from the institutions point of view must be used to control the agents' behaviour. Moreover, this enforcement mechanism cannot use or directly change the inner workings of the agent. That is, the institution can neither directly observe nor control the beliefs, desires and intentions of the agents in the system.

1.2.1 Research Questions

The considerations above lead to the main high-level problem statement.

- What is the relation between electronic institutions and legal specifications in law and regulations?

An important aspect of this question is how an institution can be designed and implemented based on a specification in (human) laws and regulations. In order to implement electronic institutions based on human laws, our idea is to use an explicit specification of the norms, and enforcing the norms from an institutional perspective. Because institutions are modelled by their norms and procedures, we also need a method for designing new procedures in case none can be used from the real world. This leads us to the following research questions.

1. How can norms be implemented from an institutional point of view?
2. How can protocols and procedures be designed for a highly-regulated domain?

Since the institution is specified by its protocols and norms, it is important that the protocols used in the institution do what they are supposed to do. Protocols of an (electronic) institution provide the members of the institution with a general manner to solve (simple) tasks. It is assumed that these protocols make sure that the task is (legally) achieved while none of the norms of the institution are broken, but can this 'norm-correctness' of protocols be guaranteed? The only way to be sure whether (new) protocols of an institution are 'safe' is to verify their norm compliance. This leads us to the following 'meta'-question.

3. How does one guarantee the norm compliance of (existing) protocols?

1.2.2 Research Objectives

In order to work out the questions above, four objectives were formulated for this research.

1. Development of a *methodology* for the design and implementation of highly regulated institutions.
2. Development of a *norm enforcement* technique that allows for the implementation of norms from an institutional point of view, with a minimal loss of autonomy of the agents.
3. Development of a *design methodology* for norm-compliant protocols and procedures that can be used in highly-regulated institutions.
4. Development of a *formal technique* to verify the norm compliance of protocols against the norms that govern the domain.

With the development of a methodology for the design and implementation of highly-regulated institutions, as is the aim of our first objective, we provide an answer to the first research question, as the methodology makes clear how the norms (extracted from the laws and regulations) will be translated and implemented in an electronic institution. As mentioned above, the implementation of the norms, as specified in this methodology, will be done by an explicit representation of the norms and the development of an enforcement mechanism to see to it that the compliance to the norms is ensured. This is our second objective, and answers the second research question. To help agents that are unable to reason about norms, our implementation uses protocols to give a guideline for (legally) achieving frequently done tasks in the institution. The design of such protocols, however, can be hard due to the many restrictions that exist for a highly normative domain, which is related to the third of our research questions. To answer this question we developed a method for designing protocols based on a normative specification, as stated in the third objective.

However, given their importance in the implementation, it is important to make sure that the protocols used do not break any of the norms, which is related to the fourth research question. To guarantee that protocols, those designed by the method of the third objective and those taken from real world practice, are in fact not breaking any of the norms, we present a formal framework based on logic to verify that a protocol is actually norm-compliant, which is the aim of our fourth research objective.

1.3 Overview

This dissertation mainly focuses on the role of norms and protocols in electronic institutions, and presents a methodology to design and implement institutions based on these elements. In highly-regulated domains, where it is necessary to comply to many different laws and regulations (e.g., the ANITA scenario), it can be hard to find the right balance between the control of the system (to guarantee the compliance to the norms) and the autonomy of the participating agents (one of the defining characteristics of intelligent agents, which truly distinguishes them from software objects and methods).

The remainder of this thesis is structured as follows.

Chapter 2 provides the setting for this thesis by a discussion of the concepts that are relevant for the design and implementation of normative systems. There we present the related research done towards the concepts of norms, agents and institutions, and give their meaning with respect to the rest of this thesis.

Chapter 3 introduces a methodology to facilitate the design and implementation of highly-regulated institutions that model norms that are derived from human laws and regulations. The primary component of this methodology is the specification of the norms, which is derived from the laws and regulations, that is used to implement the institution by means of (active) norm enforcement and protocols (see later chapters for more details on these sub processes of the methodology). Moreover, by explaining the methodology to design and implement institutions we go into detail about the relation between the human laws and the electronic institution, answering our primary research question.

Chapter 4 goes into more detail about the implementation of norms. After a closer look at the different manners to implement norms, from an agent's perspective or an institutional perspective, we set out to formalise the implementation of norms from an institutional point of view. To reduce the loss of autonomy of the agents participating in the institution, and thus achieving our second objective, we propose an active norm enforcement mechanism based on violation detection and reaction.

Chapter 5 further explains the need of designing protocols for institutions and introduces a technique, and thereby achieving our third objective, to semi-automatically design protocols for domains that are regulated by lots of abstract norms. This technique uses an intermediate level of landmarks to connect the abstract norms to the concrete protocols.

Chapter 6 introduces the formal framework for the verification of the protocols. Not all protocols used in the implementation of an electronic institution are (necessarily) designed specifically for the institution (by means of the technique of chapter 5), but can also be taken from procedures that are used in the real-world counterpart of the institution. Guaranteeing that those protocols are actually not breaking any of the norms of the system, their norm compliance has to be verified. The framework presented in this chapter, and the proof of the example protocol shown, are the answer to our fourth research question.

Chapter 7 provides a summary of the work presented in this thesis and its contributions, and proposes some of the lines of research for future work.

Chapter 2

Background: Norms, Agents and Institutions

Before we formally introduce the links between norms and institutions and present our methodology for creating normative agent systems based on a set of (human) laws and regulations, we define the meaning of these concepts and give their relation to existing research. The research on norms and normative systems is not a new field of science, but existed for many years. It also spans different disciplines, e.g., norms and social interactions have long been studied in sociology, norm creation and norm usage has been studied in legal theory. Only recently have the results from these fields of science found their way into the field of computer science, where they are, among other, used to solve coordination problems in multiagent systems. In this chapter we present an overview of all topics that are important for the design and creation of normative multiagent systems, in particular the topics of norms, agents and institutions.

Clearly norms play an important role when designing normative agent systems based on human norms. Therefore, norms play an important part in this chapter and the remainder of this thesis, and we start this chapter by a discussion of different views on norms (from different research disciplines) in section 2.1. In this section we also pay extra attention to the representation of norms, as the formalisation of norms expressed in natural language is hard and can hold many problems for the resulting system. The other important topic of this chapter (though less important than norms) is agents and agent systems. Therefore, after our discussion of norms and norm representation, in section 2.2 we look at current research on agents and multiagent systems to give an idea of the problems of coordination and cooperation that exists in the creation of these kinds of systems. Since, for the purpose of this thesis, we are more interested in the normative system itself (the environment) than the agents operating in the system, we focus a bit more on the one aspect of agents that is heavily influenced by the design of normative systems, namely autonomy. Furthermore, we show in section 2.3 how the definition of social structures can help solve the problems of coordination and cooperation in multiagent systems. Moreover, to minimise the loss of autonomy of the agents in the social structure, it is advantageous to specify the conventions and rules in the society by means of norms, thus giving rise to the idea of using institutions to control and coordinate multiagent approaches. In section 2.4 we discuss this notion of institutions and, in particular, the relation between institutions and norms and the concept of electronic institutions.

2.1 Norms

Norms have been studied in many different fields of research (ranging from ethical and moral philosophy to sociology and computer science). Norms are generally observed as abstract statements with a prescriptive meaning, rather than a descriptive one, expressing how things should or ought to be, how to value them, which things are good or bad, or which actions are right or wrong. According to sociology a norm is a rule or a standard of behaviour shared by members of a social group. According to philosophy, a norm is an authoritative rule or standard by which something is judged and, on that basis, approved or disapproved.

As said, a major characteristic of norms is that, unlike propositions, they are not descriptively true or false, as norms are not intended to describe anything, but rather prescribe, create or change something. While the truth of a descriptive statement is primarily based on its correspondence to reality, philosophers, starting with Aristotle, state that the (prescriptive) truth of a normative statement is based on its correspondence to the ‘ideal’.

The creation and use of norms is an activity that has evolved in human societies for centuries, an evolution that has been closely studied by *Legal Theory*. Where in primitive cultures no written laws existed but shared morals and conventions (*informal norms*) were used to solve disputes (sometimes aided by wise men that had a better understanding of the conventions), some cultures adopted a collection of written laws (*formal norms*) to express the morals and conventions. A good example of the latter is the Roman civilisation, which created a normative system where Law is an interpretation of codified values and norms (the earliest codification of law in Roman history is the *Lex Duodecim Tabularum*, the Twelve Tables, which contained a number of specific provisions designed to change the customary law already in existence at the time). This view, which was the origin of *Roman Law*, was later extended by Germanic kings and given a full formalisation of all different aspects of legal systems. At present day, most European countries that were part of the Roman Empire, those linked to the first through monarchical marriages, and colonies of the former, use legal systems based on this Roman-Germanic type of Law. In most of them the normative system is hierarchical; norms are stated in the *constitution*, *laws* and *regulations*, with the constitution being the highest (most important) form of norms and regulations the lowest. This layered approach allows the legal system to assess the legality of the collection of norms and rules itself. This means that regulations, that break one or more laws, can be stated to be illegal. Similarly, regulations and laws can be considered *non-constitutional* if they break one or more of the articles in the constitution.

A different example of written law, which is used, for instance, in the United Kingdom and the United States, is the *Common Law*, where law is composed of an aggregation of past decisions that are recorded for future use. In this case the law is based on a case-based approach, where decisions on a current issue are taken based on searching for similar decisions taken in the past. The highest power in the normative system used in the United Kingdom is the *Parliament*, which may decide if something is legal or illegal, either by past decisions or by taking decisions in new topics. The normative system of the United States, which is also based on Common Law, however, uses a constitution as its backbone. Situations and actions can be stated legal or illegal based on past decisions or based on the constitution. Moreover, past decisions of governors or judges can be revisited and stated legal or illegal depending on them being constitutional or non-constitutional. In this thesis we focus on systems based on Roman Law, due to the amount of work already done on the formalisation of this kind of legal systems[1].

The study of norms has lately gained attention from the field of *computer science*, including from the agent community. Norms are used there to solve coordination issues that appear in open multiagent systems (systems where the agents, which are autonomous entities[2], can join and leave at any time they choose). Given that multiagent systems have become larger and more complex in recent years, modelling complex distributed systems, that distribute information and tasks among several agents to model complex environments, a new interest was formed in views of what a norm is, how a norm can be modelled and how compliance to norms should be gained. This has given rise to new concepts such as virtual organisations and agent-mediated electronic institutions.

The main points of focus for research in the computer science community have been to formalise human normative systems, especially on the expressiveness of logical representations in order to model complex issues in human normative systems [Alchourón and Bulygin, 1971; Prakken, 2001a,b; Prakken and Sartor, 2002; Hage, 2000], but also on ontological representations of the knowledge needed for legal reasoning [Boella and Lesmo, 2000; van Engers and van der Veer, 2000], and how norms can be integrated in agents and agent structures [Dignum, 1999; López y Lopez, 1997; Castelfranchi et al., 2000; Verhagen, 2000; Vázquez-Salceda, 2004].

Later on, in section 2.2 and beyond, we look at agents, agent systems and how

[1]A discussion on the formalisation of (legal) case-based reasoning can be found in, e.g., [Rissland et al., 2003].

[2]A more descriptive explanation of Agents follows in section 2.2.

norms can be used to control and regulate agents, thus solving the coordination and cooperation problems that are inherent to open multiagent systems. First, however, we look at how norms can be formally represented and the problems that arise when attempting to formalise (real-world) norms.

2.1.1 Representations of Norms

One issue of norms, shared by legal theory and computer science, is the problem that norms and rules are usually expressed in natural language and that it can be hard to capture their precise meaning in a clear formal representation. Early representational attempts, which were based in propositional logic and later in propositional modal logic [von Wright, 1951], had to cope with many problems that lead to unwanted (or rather unintuitive) derivatives of the chosen representation (the so-called paradoxes). This first representational formalism, the *Standard Deontic Logic* (SDL) based on [von Wright, 1951], consists of the following axioms and rules (O stands for obligation, P for permission, F for prohibition and $\perp$ stands for falsum):

$(Taut)$	the tautologies of propositional logic (or just enough of them)
(K_O)	$O(p \rightarrow q) \rightarrow (Op \rightarrow Oq)$
(D_O)	$\neg O\perp$
(P)	$P(p) \leftrightarrow \neg O \neg p$
(F)	$F(p) \leftrightarrow O \neg p$
(N_O)	$p \vdash Op$
(MP)	$p, p \rightarrow q \vdash q$

SDL has a Kripke-style modal semantics based on possible worlds, where accessibility to another world (associated with the O-modality) points out the 'ideal' or 'perfect deontic alternatives' of the world under consideration. Formally, the models are like $\mathcal{M} = (S, \pi, R_O)$, where S is the set of states, π is the truth assignment function, and R_O is the deontic accessibility relation (which is assumed to be serial, thus for all $s \in S$ there is a $t \in S$ such that $R_O(s,t)$). The interpretation of the O operator would then be by means of the relation $R_O : \mathcal{M}, s \models Op$ iff $\mathcal{M}, t \models p$ for all t with $R_O(s,t)$. This system can be shown to be sound and complete with respect to validity.

The paradoxes of deontic logic, as spoken of earlier, are logical expressions that are *valid* in the logical system for deontic reasoning, but which are *counterintuitive* in a common-sense reading. Naturally there is the general problem of matching a formal approach with the 'practice', i.e., do formal theorems in logic match the intuition? This problem seems, however, to be more persistent in deontic logic than in other (modal) logics (although similar problems exist in, e.g., epistemic logic through the problems of logical omniscience, see [Meyer and van der Hoek, 1995]).

Deontic logic has its origins in Moral Philosophy and the Philosophy of Law and was traditionally used for reasoning about ethical and legal aspects. Therefore, the logic has been evaluated critically in its capability of giving an adequate representation and reasoning mechanism for these purposes. Originally, the paradoxes were also discovered and judged against this background. It remains to be seen, however, whether the paradoxes are equally problematic when deontic logic is used for the specification of advanced information systems.

Some of the most well-known paradoxes of deontic logic, these paradoxes are theorems of SDL, are the following.

1. Ross' Paradox: $Op \rightarrow O(p \vee q)$
2. No Free Choice Permission: $(Pp \vee Pq) \leftrightarrow P(p \vee q)$
3. Good Samaritan Paradox: $p \rightarrow q \vdash Op \rightarrow Oq$
4. Chisholm's Paradox: $(Op \wedge O(p \rightarrow q) \wedge (\neg p \rightarrow O\neg q) \wedge \neg p) \rightarrow \bot$

As mentioned, the strangeness of most of these paradoxes becomes more evident if they are given an informal reading. Ross' Paradox can thus be illustrated as "if one is obliged to mail a letter, one is obliged to mail the letter or burn it", which sounds odd indeed. The second paradox states that if one is permitted to do either p or q, it is equivalent to say that one has either permission to do p or permission to do q. This is in contrast with a free choice reading of $P(p \vee q)$, where it is understood that this should imply to have *both* the permission to do p *and* the permission to do q. The good Samaritan paradox, which expresses that every logical consequence of something that is obliged is obliged as well, can be illustrated best in the following reading: "if John helps Peter who had an accident" implies that "Peter had an accident", which appears to be true, it follows that "if John ought to help Peter who had an accident" implies that "Peter ought to have an accident", which certainly is not true. The last paradox, which is considered the most serious paradox in (standard) deontic logic, involves *contrary-to-duty imperatives*. Contrary-to-duty imperatives are a specification of norms in case some other norm(s) have already been violated. The paradox is best illustrated by the following statements in natural language:

(1) You ought to go to the party (Op)

(2) If you go to the party, you ought to tell you are coming $(O(p \rightarrow q))$

(3) If you do not go, you ought to not tell you are coming $(\neg p \rightarrow O\neg q)$

(4) You do not go to the party $(\neg p)$.

Intuitively, this set of statements seems to be perfectly understandable and consistent, and, moreover, none of the four statements seems to be redundant. However, the representation of these in SDL is inconsistent. This can be seen by combining K_O, $O(p \rightarrow q) \rightarrow (Op \rightarrow Oq)$, and $O(p \rightarrow q)$, which yields Oq when combined with Op. On the other hand, combining $\neg p \rightarrow O\neg q$ and $\neg p$ yields $O\neg q$, which contradicts the earlier result Oq because of D_O.

Note that we represented (2) and (3) in different ways (which, of course, may be questionable), however, while we could replace (3) with $O(\neg p \rightarrow \neg q)$ we run into problems because this is derivable from (1), which violates our initial intuition that all four statements (1) – (4) are independent. Similarly, replacing (2) by $p \rightarrow Oq$ violates this intuition as this is derivable from (4).

In an attempt to solve these problems, [Meyer et al., 1996, 1998] identifies four confusions that count for at least a part of these problems:

- confusion between *ought-to-be* and *ought-to-do*.
- confusion between the formal interpretation (of the logical operators $\wedge, \vee$ and $\rightarrow$, for example) and the natural language (commonsense) reading of these.
- confusion between ideality and actuality; and, in particular, an overestimation of ideality and the notion of perfect alternatives as it appears in the formal semantics. In particular, norms can conflict in reality.
- confusion between normative notions necessary in general abstract contexts (such as ethics) and those needed (and sufficient) for a concrete practical application.

The first confusion mentioned relates to the confusion about Op that has existed since the inception of deontic logic. This relates to the actual meaning of Op, or rather the meaning of p in this context; is it a description of a state of affairs or does it denote an action, thus expressing either an obligation-to-be (*Seinsollen*) or an obligation-to-do (*Tunsollen*)? [Meyer et al., 1998] argues that for SDL viewing p as an action makes the O-modality inadequate (SDL is not appropriate for denoting ought-to-do), however it might be adequate enough if p is considered a state of affairs, if it concerns concrete situations. More importantly, [Meyer et al., 1998] states that this should result in having distinct (and really different) logics for ought-to-be and ought-to-do. Our logic for the representation of norms, presented in chapter 4.2.1, is a logic for ought-to-be. Although we represent actions and translate norms that appear as ought-to-do, the representation of actions is, in fact, done as states of affairs.

The second confusion should be considered more as a warning. Realising that the formal interpretation of the operator does not match the commonsense one, one should just use the operators in their formal interpretation without trying to solve the mismatch. It is important to note, though, that following this course of action means that extra caution must be exerted when translating informal specifications (e.g., in natural language) to formal representations and vice versa.

The third confusion actually describes that (formally) $O\bot$ is not equivalent to $\bot$, i.e., in reality it can be that conflicting duties exist, which, of course, need to be resolved. Reasoning about one's duties in a situation with inconsistent norms, trying to solve the conflicts that arise in one's duties (*duty maintenance*), is very much related to *defeasible* (nonmonotonic) reasoning, cf. [Lukaszewicz, 1990]. This defeasible or nonmonotonic reasoning is about considering (sequences of) steps while having incomplete or inconsistent information, but still being able to draw conclusions based on (parts of) the information. Inspired by this approach, the AI & Law community rapidly developed an interest in the defeasible approach to deontic logic and legal reasoning, cf. [Jones, 1993; Meyer and Wieringa, 1993; Prakken, 1993, 1994; Verheij, 1996; van der Torre, 1994].

The last confusion is an important one for solving the paradoxes mentioned above in the context of building a practical application, as is the goal of this thesis. When specifying concrete systems one is mostly interested in different problems than when one is analysing deep philosophical problems. Specifying normative systems require a more *pragmatic* view on matters involved with the reasoning about norms. [Meyer et al., 1998] states that the most important role of deontic logic, when used for normative system specifications, is to accommodate for the need to distinguish between *ideal* and *actual* behaviour of the system [Jones and Sergot, 1992]. In this case, one would use Op (as an ought-to-be) to mean 'p is desirable' or 'ideally p is the case'. This gives us the option to specify what is to be done if the actual behaviour deviates from the ideal or desired behaviour. Moreover, using Op as 'ideally p' allows even SDL (as well as Anderson's variant of SDL, on which we base our logic formalism in chapter 4.2.1) to be useful, since all paradoxes but Chisholm's become perfectly understandable and intuitive properties of 'ideality'. It then remains to provide a (pragmatic) solution to the Chisholm paradox.

The solution to the Chisholm paradox, as given in [Meyer et al., 1998], consists of making a difference between the violations that can occur in the paradox, which is done by using multiple obligations. The logic $S5O_{(n)}$ used is a modal logic with the modalities of universal necessity $\Box$ and a number of distinct obligations O_i to express obligations with respect to a frame of reference i. The system consists of the following axioms ($\Diamond$ is used as an abbreviation of $\neg\Box\neg$):

$(K_\Box)$	$\Box(p \to q) \to (\Box p \to \Box q)$
$(T_\Box)$	$\Box p \to p$
$(5_\Box)$	$\Diamond p \to \Box\Diamond p$
(K_i)	$O_i(p \to q) \to (O_i p \to O_i q)$
(D_i)	$\neg O_i \bot$
(P_i)	$P_i p \leftrightarrow \neg O_i \neg p$
(F_i)	$F_i p \leftrightarrow O_i \neg p$
$(\to)$	$\Box p \to O_i p$
$(Taut)$	the tautologies of propositional logic (or just enough of them)
(MP)	$p, p \to q \vdash q$
$(N_\Box)$	$p \vdash \Box p$
(N_i)	$p \vdash O_i p$

The semantics are similar to the semantics of SDL, with the extension of additional accessibility relations: $\mathcal{M} = (S, \pi, R, \{R_i \mid i = 1, \ldots, n\})$, with S the set of states, π the truth assignment function, $R = S \times S$ the (universal) possibility relation, and R_i the deontic accessibility relations (again, assumed to be serial).

The operators $\Box$ and O_i are interpreted by means of their, respective, accessibility relations: $\mathcal{M}, s \vDash \Box p$ iff $\mathcal{M}, t \vDash p$ for all t with $R(s,t)$; and $\mathcal{M}, s \vDash O_i p$ iff $\mathcal{M}, t \vDash p$ for all t with $R_i(s,t)$.

The most important feature of this system is the *non-validity* of $\neg(O_i p \wedge O_j \neg p)$ (for $i \neq j$), which expresses that it is indeed possible to represent contrary norms in a consistent manner, by using distinct frames of reference, even though within a single frame of reference i $O_i p \wedge O_i \neg p$ is still inconsistent.

The Chisholm paradox can now be represented as

(1′) $O_1 p$

(2′) $\Box(p \rightarrow O_2 q)$

(3′) $\Box(\neg p \rightarrow O_3 \neg q)$

(4′) $\neg p$

From these we can derive that $(3') : \Box(\neg p \rightarrow O_3 \neg q) \Rightarrow \neg p \rightarrow O_3 \neg q \Rightarrow_{(4')} O_3 \neg q$ and $(2') : \Box(p \rightarrow O_2 q) \Rightarrow O_1(p \rightarrow O_2 q) \Rightarrow O_1 p \rightarrow O_1 O_2 q \Rightarrow_{(1')} O_1 O_2 q$. This gives a clear record of how contrary norms are handled (either in frame 3, or in frame 2 via frame 1) without being inconsistent. The logic we present in chapter 4.2.1 uses a similar method of distinguishing between different obligations to solve the Chisholm paradox.

It is worth noting, however, that all attempts to represent norms, i.e., in SDL or later developed deontic logics based on dynamic logic [Meyer, 1988]or temporal logic [Gabbay, 1976; van Eck, 1982], have similar problems as shown here for SDL. While the presented paradoxes might be solved in more expressive formalisms, other paradoxes might be found which could be just as hard to solve. Unfortunately, there does not exist a single deontic representing formalism that is entirely free of representational problems, but, as already mentioned above, to deal with these problematic cases of representation one may adhere a pragmatic view and use logics that are *adequate* for concrete situations. Even though such a logic might not always be adequate in general, one need not solve all profound problems that philosophy poses for the general abstract cases [Meyer et al., 1998].

2.2 Agents

Agent-based systems are considered one of the most promising new areas of research and development to have emerged from the field of information technology. Because the focus of research in the field of computer science shifted towards distributed and dynamic systems, agent-based approaches have been used more and more as the solution to issues pertaining to coordination and effectiveness in open, distributed and dynamic systems, covering domains such as electronic commerce, health care and resource allocation. Agents (the word *agent* comes from the Latin verb *agere* and applied to software, it means ‘something that acts’) are a new paradigm for developing software applications and made it possible to support the representation, coordination and cooperation between heterogeneous processes and their users.

While many different definitions of agents exist, the most common definition of an agent used, as given in [Wooldridge and Jennings, 1995], is that 'an agent is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives.' Here it is worth noticing that 'encapsulated computer systems' signifies a distinction between the agent and their environment, but also implies that there exists a well-defined boundary and concrete interface between the agent and its environment. The key aspect of the definition, however, is *autonomy*, which signifies the important difference between agents and software objects; objects have no control over its own methods (once a publicly accessible method of the object is called, the corresponding actions are performed), but agents can decide on their own whether to perform a requested action [Wooldridge, 1997].

Given their study in [Wooldridge and Jennings, 1995], Wooldridge and Jennings identify the following four attributes as most commonly ascribed to agents.

- *Autonomy*: agents are self-regulating, guiding their actions according to their own internal rules, and capable of taking its own decisions with no external control.
- *Reactive*: agents perceive aspects of the environment and respond to changes accordingly.
- *Proactive*: agents can take the initiative in performing goal-directed actions.
- *Social*: agents communicate with other agents, including humans, not only to share information, but also coordinate actions to achieve goals that are unreachable for one agent on its own, thereby being able to solve even more complex problems.

Other properties claimed for various agents include mobility, rationality, intentionality, desire, commitment, deliberation, flexibility, and robustness. Because of the significance of the aspect of autonomy to the topic of this thesis we discuss this in more detail in subsection 2.2.1.

Multiagent systems (MAS) are slowly becoming the leading option for the implementation of open distributed systems in complex domains, mainly because of their ability to distribute the work over several autonomous, rational and proactive entities and to adapt and reorganise itself when needed. Building multiagent systems, however, is a very complex process. The simplest of these are the *closed multiagent systems*, where the designer (or a team of designers) programs all the agents that will cooperate in the system himself to ensure the cooperation but reduce the complexity of the development. The design process can thus be seen as making a cooperative problem solving system, which is driven by the definition of a set of common goals. These common goals, also called global goals or system goals, are used to guide the system to the desired solution. Although goals from individual agents in the system may differ to some extent, the agents do agree on the global goals and include them in some way in their set of individual goals. Other advantages of closed multiagent systems, besides this predefined goal-oriented cooperation, are the simple solution to communication problems (all agents are developed with the same ontology) and the assurance that the multiagent system covers all tasks (it is known, at design time, whether there will be agents available for completing all aspects of the tasks given).

Open multiagent systems, on the other hand, are more challenging to design. These kinds of systems allow heterogeneous agents, developed by different designers, to join and leave the system at all times. In such scenarios agents can have completely different goals, and even have goals that conflict with goals of other agents. Unlike the closed multiagent systems, where the inner workings of the other agents is, implicitly, known to the agents in the system, no guarantees can be given about other agents in an open system, making *trust* a big issue. Trust, as defined in [Gambetta, 1990], is a particular level of subjective probability with which an agent will perform a particular action before the action is actually performed. In an open system trust is not easy to achieve, since agents may give incomplete or false information to other agents or betray them if such actions allow them to fulfil their own individual goals. In fact, due to the heterogeneity of the agents, and the anonymity caused by having all interactions among the participants made over distance (opposed to closed multiagent systems, where all agents can be run on the same computer or on a private network, open systems are generally run over public communication channels, such as the internet) trust among the participants is very difficult to obtain. Unlike the closed systems, where trust is implicit, in open agent systems agents can join that intentionally betray other agents for their personal gain. The solution to this problem is the introduction of some form of control over the agents in the system to guarantee a certain level of safety, thus inspiring trust in the agents. Naturally, introducing control mechanisms in a system limits the possibilities of the participants of the system, i.e., it reduces their autonomy. The problem of open system design then becomes to find the correct balance between the control of the system versus the autonomy of the agents participating.

2.2.1 Autonomy

Before we go into detail about control mechanisms used in open agent systems, and analysing the control mechanisms used in real world societies, let us first have a closer look at the concept of *autonomy*. Autonomy, as defined in [Wooldridge and Jennings, 1995], is the capability of an agent to act independently, exhibiting control over its own internal state. It can be argued to be the main characteristic of agents, but it is definitely one of the defining characteristics of an agent. A different definition of autonomy, in [Verhagen, 2000] which is complementary to the one above, is that autonomy is the degree in which one agent is independent with respect to another entity, the environment, other agents or even its own developer. Verhagen notes that autonomy should be considered both at the individual level as well as at the MAS level. On the individual level one can identify the following levels of autonomy of agents: 1) *autonomy from stimuli*, which is the degree to which the behaviour of the agent is controlled by external stimuli received from other entities; 2) *cognitive autonomy*, which is the degree to which the (internal) choices of the agent are governed by other agents; and 3) *autonomy of norms*, which is the degree to which an agent is capable of learning new rules of behaviour, thus making them able to diverge from predefined decision making rules. On the level of the whole system, Verhagen identifies two different types of autonomy: 1) *norm autonomy*, which is the degree to which social influences govern the choices of the agents, and 2) *designer autonomy* which refers to the autonomy of the agents forming a MAS with respect to the self-organising capabilities of a MAS, which may lead to emergent behaviour.

Although differences exist between autonomy at the individual level and at the MAS level, they are not entirely independent and overlap exists between them. A more accurate definition of autonomy hierarchy is given by Verhagen, through the proposal of the following four levels of autonomous agents.

- *Reactive Agents*: agents whose autonomy completely resides in the combination of environmental cues and system properties. The perception of the agent is directly coupled to action, and there exists no autonomy on the individual level, only the autonomy of the MAS as a whole.

- *Plan Autonomous Agents*: agents that are autonomous in their choosing of the sequences of actions (plans) to obtain goals. The goals itself are inherent to the agent or triggered by requests from other agents. That means that the autonomy of the individual agents is reduced to only the choice of actions to achieve the externally imposed goal.

- *Goal Autonomous Agents*: agents that are autonomous in making decisions about goals, enabling them to choose their 'prevailing interest', with respect to its goals. The agent is autonomous in determining which states of the world are desired, given goal satisfaction and goal priority.

- *Norm Autonomous Agents*: agents with the capabilities to choose goals that are legitimate to pursue, based on the norms of the system. Moreover, norm autonomous agents are equipped to judge the legitimacy of their own and other agents' goals. Verhagen defines autonomy at this level as the agent's capability to change its norm system when a goal conflict arises, thereby changing the priority of goals, abandoning a goal, or generating a new goal. In [Dignum, 1999] another view of norm autonomy is given, where the agents are allowed to violate a norm in order to adhere to a private goal that they consider to be more profitable, while taking the negative value of the repercussions of such a violation in consideration.

It is important to note, however, that even at the highest level of autonomy, the agents of a MAS will never have complete autonomy, because the agents in a MAS will never have sufficient competence, resources or information to solve the entire problem. Because of their limited capabilities to handle only some of the tasks they are always dependent on other agents in the MAS [Jennings, 1993].

2.3 Social Structures

As we mentioned earlier, an important problem in the creation of a MAS is the coordination between the agents. This coordination consists of a set of mechanisms necessary for the effective operation of a MAS in order to get a well-balanced division of labour, while reducing the logical coupling and resource dependencies of agents. Lots of research has been done towards the study of coordination, both for specific domains as well as more generic, domain-independent views, which lead to the creation of coordination theories (examples of coordination theory include joint-intentions theory [Cohen and Levesque, 1991], theories about shared plans [Grosz and Kraus, 1996], and teamwork models [Jennings, 1995; Tambe, 1997]).

One way of solving the coordination complexity of a MAS is by creating a central controller (typically a specific agent) that ensures the coordination between the agents in the system. These coordinator agents assign the work to the individual agents in the system, based on their capabilities. While this approach greatly reduces the complexity of the system, as the ultimate goal of the system is ensured by the goals of the coordinator, this reduction is not without a cost. The main disadvantages of this approach, which stem from the centralisation of control, are a loss of robustness and flexibility, since the coordinator agent becomes a vital part of the system, creating a bottleneck and a liability (if the coordinator agent becomes overloaded or crashes, it breaks the entire system), and a severe reduction of the autonomy of all agents other than the coordinator.

An alternative is to distribute both the work and the control among all the agents in the system. In that case, all the agents need to be equipped with reasoning and social abilities to be able to decide on the intentions and knowledge of the other agents, in order to successfully coordinate and resolve problems when they arise. However, in domains where conflicts are best avoided (the cost of a conflict would be too severe or conflict resolution is hard), complete independence of behaviour is unwanted, [Moses and Tennenholtz, 1995]. To solve this problem, some kind of structure should be defined to ease coordination in the system, for instance defining *social structures*.

A social structure, which defines a social layer for a MAS, where the MAS is seen as a society of entities, defines structured patterns of behaviour in order to enhance the coordination of agent activities. By restricting the agents' actions, social structures reduce the complexity of dealing with agent cognition, cooperation and control, as it makes it easier for a given agent to predict and model what other agents would do in a given situation and adapt their own behaviour accordingly.

Social structures are classified by [Findler and Malyankar, 2000] into 5 groups:

- *alliances* are temporary groups formed voluntarily by agents whose goals are similar enough. While in the alliance, the agents give up some of their individual goals and fully cooperate with other members of the alliance, but stay only as long in an alliance as long as it is in their interest;
- *teams* are groups formed by a special agent, the team leader, who recruits qualified members to solve a given problem;
- *coalitions* are similar to alliances but members do not have to abandon their individual goals but only participate in those joint activities which goals are not conflicting with their own goals;
- *conventions* are the formal description of preferred and prohibited goals and actions in a given group of agents; and
- *markets* are structures where two prominent roles, buyer and seller, define the mechanisms for transaction interactions.

In addition to these groups of social structures, the structure of a group of socially situated agents can, in some cases, emerge from the interaction between the agents, instead of being specified beforehand. This kind of emerging social structures is modelled in *referral networks* [Yu and Singh, 2002].

A more generic description of social structures is proposed in [Dignum, 2004], where an organisational theory point of view is taken rather than a sociological classification as done above. In this proposal social structures are divided in the following three groups:

- *markets*: agents are self-interested, driven completely by their individual goals. Interaction in markets occurs through communication and negotiation;
- *networks*: coalitions of self-interested agents that agree to collaborate in order to achieve a mutual goal. Coordination is achieved by mutual interest, possibly using trusted third parties; and
- *hierarchies*: agents are (almost) fully cooperative, and achieve coordination through command and control lines.

The organisational theory point of view of this categorisation gives it two advantages (also see table 2.1). First, it is more generic than the approach mentioned earlier because it considers both human and software agent organisations, thus being able to capture all 5 categories mentioned in the approach by Findler et al. Next to that, although being generic, it is very useful for designing organisations as it tries to motivate the choice of one of the structures by their appropriateness for a specific environment. It identifies for each of the groups the kind of problems that are solved best, as well as giving some of the facilitation tasks that should be provided by the social structure.

	MARKET	**NETWORK**	**HIERARCHY**
Coordination	Price mechanism	Collaboration	Supervision
Relation form	Competition	Mutual interest	Authority
Primary means of communication	Prices	Relationships	Routines
Tone or Climate	Precision/suspicion	Open-ended/ mutual benefits	Formal/bureaucratic
Range of cooperation	No cooperation expected	Negotiation of cooperation	Absolute cooperation expected
Conflict Resolution	Haggling (Resort to courts)	Reciprocity (Reputation)	Supervision

Table 2.1: *Organisational forms according to [Dignum, 2004].*

In [Jennings, 1993] an alternative view of existing coordination models is presented and an attempt is made to create an unifying framework based on the concepts of *commitments* and *conventions*. Commitments are defined as the promises to undertake a specific course of action, while conventions are given as the means to monitor commitments in changing circumstances. It is then stated that all coordination mechanisms can be reformulated by means of (or reduced to) (joint) commitments and their related (social) conventions.

An important issue raised by this approach is to find out how conventions are to be introduced in the interactions of the agents. Although it is possible to hard-code the conventions into the agents as restrictions, i.e., reducing the set of possible options at the decision level of the agents, doing so would be a severe reduction of the autonomy of the agents. Another approach (as currently done, for instance, by [Esteva et al., 2004a]) would be to reduce the set of options on the protocol level. In this way the conventions are encoded into the protocols that the agents follow to ensure the coordination, completely defining the ways of reacting to received messages. Although this scenario seems acceptable, having the conventions encoded in the protocol restrains the agents from deciding to violate the convention, leading to, as shown in [Dignum, 1999], *loss of autonomy*, since the agents always follow the protocol they react to messages in a predictable way which can be exploited to gain control over the agent, and *loss of adaptability*, since fixed protocols do not have the capacity to react to (unforeseen) changes in the environment.

In [Dignum et al., 2000] it is proposed to use norms as an explicit influence on an agent's behaviour. This gives rise to an interesting question in the encoding of conventions in a society of agents, namely by regulating interactions between agents by means of explicit norms.

2.4 Institutions

All human institutions are governed by conventions and rules of some sort, which define the interactions in the society. These conventions and rules either emerged from the society (they originated from the normal practice in the society) or originate from laws that were developed by the society. In this way, all human societies, even the primitive ones, use social constraints to regulate relations among its members. These constraints are either informal, for instance customs and traditions, or formally defined, like laws and regulations. The constraints of a society are adopted by its members, influencing their actions, thus creating patterns of expected behaviour.

A collection of these social constraints, which are referred to as *institutions* by [North, 1990], affect human organisations by shaping choices and making outcomes foreseeable. These constraints allow organisations and the interactions required in the organisation to grow in complexity without increasing the costs of the interactions (by ensuring trust among participants the cost of interactions is effectively reduced). The integration of institutional constraints allows every competent participant to act in the institution according to a list of rights, duties, and interaction protocols. Moreover, the institutional constraints allow participants to expect others to act in similar manners.

The main reason for creating institutions is to build trust among parties. In those environments where little to no information is available about other participants (like global commerce), trust is needed to establish coordination and cooperation between parties. The creation of an institution, in such a domain, provides sufficient information for all individuals to create trust by giving constraints (defining the behaviour that can be expected from others) and regulating violations of those constraints.

2.4.1 Institutions and Norms

Most of the descriptions of institutions are based in the formal definition of their norms and the effect of the norms on the actions that are performed by the participants of the institution. These aspects of norms in societies (of agents) have been studied by sociology.

Sociology, a study that evolved from moral and ethical philosophy (which itself dates back to ancient Greece and Rome, as found, for instance, in Plato's moral philosophy and Aristotle's study of ethics), aims to describe the interactions that arise among members of groups, and the social structures that might be established. The modern approaches of sociology focus on open systems, where the relations between individuals are less clearly defined and established, but where these relations are largely influenced by the context or environment of the society. In these open systems, similar to open multiagent systems, problems of *social order*, such as consensus or limited trust, affect coordination and cooperation efforts made by the participants of the system. Human societies have mainly coped with such problems by the development of norms and conventions to specify the expected behaviour of the participants. An important distinction in norms in the field of sociology is shown in [Tuomela, 1995], which makes a difference between *rules* and *norms*. Rules (called *r-norms* in [Tuomela, 1995]) are created by an authority and are based on agreement-making, while norms (called *s-norms*) are based on mutual belief, and are more similar to conventions.

The real study of the use of norms in societies comes from the field of Institutional Theory that emerged from a combination of sociology and economical studies [North, 1990; Scott, 2001]. Institutional Theory is based on the fact that norms are supported by social institutions (societies that enforce the compliance to the norms on its members), and focuses on the dynamics of regulatory frameworks (norm emergence, norm imposing and norm spreading), and the effects of regulatory frameworks on the dynamics of a social system and its participants.

In [North, 1990], for instance, a study is presented of the effects of normative systems on the performance of organisations and societies. North comes to the conclusion that institutional constraints, which provide trust between parties and help shape choices, improve human interactions and reduce the cost of these interactions. The establishment of institutional constraints allows individuals to behave, and expect others to behave, according to the norms.

The work presented in [Scott, 2001], which is complementary to North's as he focuses on the social aspects of institutions, defines a unifying framework to describe the interactions between organisations and institutions. He states the following distinction between rules and norms:

- *rules*: meant for describing the *regulative aspects* which constraint and regulate behaviour;
- *norms*: meant for describing *prescriptive*, *evaluative* and *obligatory aspects* of social life.

The framework presented in [Scott, 2001], summarised in what Scott calls the *Three Pillars of Institutions* (see the table below), states that institutions are composed of regulative, normative and cultural-cognitive aspects (the latter being shared views, common beliefs, etcetera), where rules are related to the legal aspects, and norms are related to the moral aspects of institutions.

A main concept found in complex social structures is *role*. A role is a description of the duties and restrictions to be adhered to by an entity. Roles have been extensively studied in Organisational Theory, in order to study the relationships among the social roles an individual may enact, the obligations and authorisations in the distribution of labour mechanisms. An interesting view on the relation between norms and roles is given by [Scott, 2001].

> "Some values and norms are applicable to all members of the collectivity; others apply only to selected types of actors or positions. The latter gives rise to roles: conceptions of appropriate goals and

	Regulative pillar	**Normative pillar**	**Cultural-Cognitive pillar**
Basis of compliance	Expedience	Social Obligation	Taken-for-grantedness Shared understanding
Mechanisms	Coercive	Normative	Mimetic
Elements	Rules, Laws, Sanctions	Values, Expectations	Common beliefs, Shared logics of action, Categories, Typifications
Basis of legitimacy	Legally sanctioned	Morally governed	Comprehensible, Recognisable, Cultural supported
Routines	Protocols, Standard operating procedures	Jobs, Roles, Obedience to duty	Scripts

Table 2.2: *Scott's Three Pillars of Institutions [Scott, 2001].*

activities for particular individuals or specified social positions. These beliefs are not simply anticipations or predictions but prescriptions – normative expectations – of how the specified actors are supposed to behave."

A further elaboration on the use of roles is given in chapter 4.2.1.

2.4.2 Electronic Institutions

As mentioned earlier, while trust being implicit in closed systems, open multiagent systems require that trust is built by some mechanism. An option would be to create institutions for these open environments, thus creating an *electronic institution*. An electronic institution is the model of a (human) institution through the specification of its norms in some suitable formalism(s). While many human institutions evolved through millennia of interactions, the goal of building electronic institutions is to create frameworks to formally design and implement agent architectures where the interactions are ruled by some kind of institutional structure.

Through the formal specification of the norms and protocols of an institution, electronic institutions provide the following advantages to a MAS:

- reduction of uncertainty about other participants;
- reduction of misunderstandings, since there is a shared collection of norms governing interactions;
- foreseeable outcomes of interactions among participants;

- simplification of the decision-making process of each agents, because of the reduction of the number of possible actions.

The main observation is that electronic institutions solve the major problems of open agent societies because they specify norms that define what is right (legal) and wrong (illegal) in the society and specify the rights and responsibilities of its members. This helps to build trust among the agents and allows the agents to plan their actions based on the behaviour that can be expected from the other participants of the institution, thus improving the achievement of their (socially accepted) goals.

The only tool currently available for formally specifying electronic institutions is the ISLANDER formalism, [Esteva et al., 2001]. This formalism provides a formal framework for institutions [Rodriguez, 2001] and has proven to be well-suited to model practical applications (e.g. electronic auction houses). It views an agent-based institution as a *dialogical system* where all the interactions inside the institution are a composition of multiple dialogic activities (message exchanges). The messages (or *illocutions* in these interactions [Noriega, 1997]) are structured through agent group meetings called *scenes* that follow well-defined protocols. This division of all the possible interactions among agents in scenes allows a modular design of the system, following the idea of other software modular design methodologies such as Modular Programming or Object Oriented Programming. A second key element of the ISLANDER formalism is the notion of an agent's *role*. Each agent can be associated to one or more roles, and these roles define the scenes the agent can enter and the protocols it should follow. Finally, this formalism defines a graphical notation that not only allows obtaining visual representations of scenes and protocols but is also very helpful while developing the final system, as this representation can be seen as a blueprint.

ISLANDER has been mainly used in *e*Commerce scenarios, and was used to model and implement an electronic Auction house (the *Fish market*). Furthermore, the AMELI platform [Esteva et al., 2004a] allows the execution of electronic institutions, based on the rules provided by ISLANDER specifications, wherein external agents may participate. The activity of these agents is, however, constrained by a special interface to the institution that regulates agent actions to the precise enactment of the roles specified in the institution model.

The ISLANDER approach to creating institutions starts by defining the interaction patterns, only to add the norms at a later stage, which works very well for institutions that are rich on procedures but are governed less by explicit norms (most of the norms are implicit in the protocols used, without an explicit representation), however, this approach is less suited for highly regulated domains with lots of restricting norms, such as in the ANITA domain. Our approach for designing electronic institutions for highly regulated domains, presented in chapter 3, is different from this as we start at the norms and add the interaction patterns in the last stage of the process.

2.4.3 Implementing e-Institutions

The creation of institutions by starting at the norms, however, has to cope with the gap that exists between normative specifications and procedural systems. The norms (used for the specification of the institution) are expressed in an abstract and vague manner and contain many concepts that have no clear meaning in the (agent) practice. If an implementation of institutions specified by a (large) number of vague norms is attempted, a number of translations has to be made to connect the concepts of the norms to concepts that are used in practice [Dignum, 2002]. This approach, as first introduced in [Dignum, 2002], creates a layered pattern for the implementation of the norms, as the abstract norms, used for the specification of the institution, are translated to concrete instantiations, which are then translated to indicate how they should be implemented (by creating direct relations to the implementation of the agent society).

This idea of a layered design for electronic institutions is also used in [Vázquez-Salceda, 2004]. There, a layered framework is proposed for the translation of institutional norms to the rules and procedures that are to be used in practice. The framework distinguishes four different levels: an *abstract level* (which contains the abstract norms), a *concrete level* (containing the concrete instantiations of the abstract norms), a *rule level* (where concrete norms are translated into rules that can be computed by the agents), and a *procedure level* (where the final mechanisms to follow the rules are implemented). The approach of [Vázquez-Salceda, 2004] uses this layered design to model institutions and gives the relations between the different layers. The methodology that we present in the next chapter uses many of the ideas presented in [Vázquez-Salceda, 2004], and goes into more details about the relations between laws and electronic institutions and mainly focuses on the process of implementing norm enforcement and designing protocols for institutions.

It is worth noting that similar manners of layered design have existed for some time in the field of *requirements engineering* [Wieringa, 1996]. Requirements engineering, also requirement analysis, is a technique used in systems engineering and software engineering that encompasses all of the tasks that go into the instigation, scoping and definition of a new or altered system. The process of requirements engineering consists of *needs analysis* (establishes why a system should exist, produces a specification of the objectives that the system must satisfy) and *behaviour specification* (establishes what a system should do, produces a specification of the observable behaviour of the system that would satisfy the stated objectives). The different layers, or levels, in this domain represent the decomposition of a system, where a system on one level is decomposed into systems at the next lower level (the systems of the lower level are said to *implement* the system at the higher level). This is called the *aggregation* or *implementation hierarchy.* The specification of *what* the system must do at a particular level of aggregation is called a *requirements specification*, and the specification of *how* the system is realised internally at the next lower level is called the *implementation.* It is then the case that the *implementation* of a system on a particular level of aggregation contains the *requirements* for the systems on the next lower level.

In system development, the generic objective of any product is to answer the needs that exist in its environment. The first step in any development process is the statement of product objectives and producing behaviour specifications and product decompositions (specifying the requirements at the highest level of aggregation, and giving the relation to the next lower level, respectively). The system decomposition on one level then represents part of the objectives of the next lower level of aggregation. This means that at each level below that of the top level objectives, the behaviour specification and decomposition specification are the objectives for systems at the next level of aggregation. This process of relating the decomposition of a system to the objectives (which can also be seen as constraints that the product must satisfy) on a lower level of aggregation is, in essence, very similar to the translation of norms from abstract specifications to concrete specifications and ultimately procedures as mentioned above. In the process of moving to a lower level, the 'vagueness' of the objectives/constraints is reduced and it can be determined easier how the objective/constraint is to be satisfied.

A second resemblance between requirements engineering and norms is that both seem to specify the required behaviour of (sub) systems. This behaviour specification is done in requirements engineering by means of the specification of properties that a system must fulfill. Requirements engineering distinguishes between two different property specifications: properties that are specified *behaviourally* and properties that are specified *nonbehaviourally* (sometimes expressed as *functional* and *nonfunctional* properties, respectively). The former express properties that are directly observable, i.e., an experiment can be specified which unambiguously tells us whether the system has said property. The latter, on the other hand, express properties of which can be agreed that a system has it or lacks it, but of which no experimental evidence exists that unambiguously settles its existence or absence in a system. Properties in this category are, for instance, the wide agreement among people about the beauty of Beethoven's sonata, or the quality of Rembrandt's paintings. In the development of a system, starting at the abstract level of a *product idea*, most properties expressing the desired behaviour of the system will be specified nonbehaviourally. Only later on when more concrete levels of development are reached (e.g., *function specification* or *transaction specification*), can most required product properties be specified behaviourally. Even then, if no behavioural specification can be found, a *proxy*[3] should be used whose presence can be specified behaviourally and which suggests the presence of the desired property.

The implementation of norms works in a similar way, where some concepts in the norms used to describe the (un)wanted behaviour (e.g., action and state

[3]Proxies are the, behaviourally specified, properties that indicate the presence of a nonbehaviourally specified property, but which are not equal to it.

descriptions) can be easily linked to observable and checkable concepts used in the agent system[4]. Other, vague and abstract, concepts, however, such as ‘fairness’, are not easily linked to concrete concepts and need to be translated to concepts that indicate the existence of ‘fairness’, but do not necessarily express the exact same meaning.

A big difference, however, between requirements engineering and norm implementations is that, due to its position in the development cycle of systems, requirements change over time while (most) norms do not. As the process of system development is a continuous cycle, where the resulting product is tested against the needs that exist in the environment, requirements that existed may have become obsolete and new requirements may have emerged. This is even more evident if one considers the fact that the product changes the environment, thus also changing the needs that exist in the environment. Naturally, the implementation of norms should be subjected to a similar process of verification, where it is checked that the resulting agent system actually adheres to the constraints that were posed by the norms. This verification, however, normally does not change the norms itself, but rather changes the implementation of the norms.

2.5 Chapter Conclusions

In this chapter we have discussed the aspects related to normative systems from different points of view. We have stated that there has been a considerable amount of research done towards the notion of norms, from different fields of research.

We have debated that the representation of norms is not without its problems, which is an issue that has existed in both the fields of legal theory and computer science, and that all representation formalisms are plagued by shortcomings resulting in derivable formulas that go against the intuitions about norms. Some of these can be quite severe, e.g., Chisholm’s Paradox, and need careful consideration to be solved, while most others can be solved by the simple pragmatic view that the representation formalism, although not capable of solving all philosophical problems, is sufficiently adequate to represent norms for usage in a concrete system (where the deep philosophical issues do not matter).

We have also argued that, inspired by the workings of human institutions and societies (studied in, for example, sociology and institution theory), norms can be an answer to the persisting problem of coordination and cooperation in open multiagent systems, where agents (entities that autonomously act in and react to the environment) can join and leave when they want and little is known about the inner workings of the other agents in the system. These open agent societies show a striking resemblance to their human counterpart in the fact that both need adequate levels of trust built among its members before being able to engage in complex interactions. Trust in these open systems, where participants may join

[4]A deeper analysis of the concepts observable and checkable, and their relation to the implementation of norms is presented in subsection 4.2.2.

that intentionally betray others for personal gain, is hard to achieve and some sort of control mechanism is required to guarantee a certain level of safety. Since these control mechanisms limit the possibilities of the participants, the biggest problem of open system design comes down to finding the right balance between the level of control of the system and the level of autonomy of the agents.

In order to reduce the loss of autonomy of the participants of the system, a control mechanism can be designed based on norms, where norms are used as an explicit influence on the behaviour of the participants. This has given rise to the use of electronic institutions, which regulate the interactions between agents in the system by means of explicit norms. The specification of norms, which define what is right (legal) and wrong (illegal) in the agent system helps to build trust and allows the agents to plan their actions based on the behaviour that is expected from them and from the other agents in the system. In the next chapter we go into more details about the relations between institutions, as a specification of norms, and the agent practice. Additionally, we introduce a methodology for the design and implementation of electronic institutions based on human laws and regulations.

Chapter 3

Normative Multiagent Systems

A methodological framework, based on previous work done on the implementation of agent institutions, is proposed to facilitate the design and implementation of highly-regulated institutions that model norms that are derived from human laws and regulations. This framework uses and enhances elements of existing institution and organisation methodologies. The framework is composed of the elementary components that an agent institution needs to be able to implement a highly-regulated environment for agents to work in. In creating this environment, we do not assume any norm-awareness of the agents in the system (though norm-aware agents would have an advantage when acting in such a normative domain), and thus need to create an environment that can handle agents that are ignorant about the norms and incapable of reasoning about norms. The main elements of the framework are; 1) an ontology to allow communication between agents, and to express the meaning of the concepts used in the norms; 2) a normative description of the domain, specifying the allowed interactions in the institution; 3) a set of protocols that agents that are incapable of normative reasoning can use to perform their assigned tasks; and 4) an active norm enforcement to see to it that the norms specified for the domain are adhered to and order and safety is guaranteed in the system.

3.1 From Law to e-Institutions

Given a specification in law and regulations, the main question then becomes how the norms specified in the law and regulations can be incorporated in the structure of an institution such that the agents operating within the institution will operate according to these norms or can be punished when they are violating the norms. While there already has been many attempts to formalise norms (e.g., [Sergot et al., 1986; McCarthy, 2002; De Vey Mestdagh, 1997; Visser et al., 1997; Tan et al., 1998]), such formalisations do not give any indications of how the laws and regulations should be interpreted within a certain institution. The concepts used in the formalisation of the norms are usually vague and abstract, and, more importantly, lack a clear connection to elements of the institutional practice. For example, a norm expressing that it is forbidden to discriminate based on age is formalised in deontic logic as '$F(discriminate(x, y, age))$'. The concept *discriminate* is, however, unclear in the institution, as no *discriminate* action exists, nor will the agents in the institution explicitly express that they are discriminating. The action states something more abstract that needs to be translated before being used in the institution [Dignum, 2002].

Because of this gap between the abstract level of the norms derived from law and the concrete level of the institution, a 'translation' between these two levels must be made before it can be determined how the norms affect the institution. This translation is highly context-dependent (based on the domain of the institution), and thus makes use of the ontology of the institution. The concrete norms resulting from this translation still have no direct relation to the implementation of the institution; a second translation is needed to indicate how the norms are implemented in the institution [Dignum, 2002]. Choices can be made on how these concrete norms are enforced, implementing them either through the introduction of constraints on the behaviour of the agents or by implementing a norm enforcement mechanism (or some combination of both).

The process of creating an institution based on a specification in laws and regulations, given the idea that institutions are characterised by their rules and conventions (see chapter 2.4), then becomes as depicted in figure 3.1.

The process of creating an institution from a specification in laws and regulations (as shown in figure 3.1) is the following. First, a formal representation of the law is created, giving an *abstract normative specification* of the allowed interactions in the institution (e.g., in deontic logic) and a basis for the ontology that is needed (we call this basis the *normative ontology*). The normative ontology built from the concepts and relations used in the formalisation of the norms is strengthened with ontological definitions given by the law itself, and information that is added from the practice (and common sense) to create the ontology for the institution (the *institutional ontology*). The normative specification is also used as the basis of the design of the enforcement and the protocols that characterise the institution. The process from norm specification to implemented norms (regimenting constraints on the behaviour of agents and norm enforcement mechanisms through violation detection and punishments) is as described above: 1) the abstract norms are translated to concrete *operational norms* (though only useable for a certain context, i.e., the institution, and less expressive than abstract norms, concrete norms are a lot easier to implement); 2) the operational norms are translated to constraints and procedures that will see to it that the norm is enforced in the institution. The design of *protocols* that can be used in the institution consists of the following steps: 1) the important characteristics of the norms that express how interactions should be in the institution are extracted

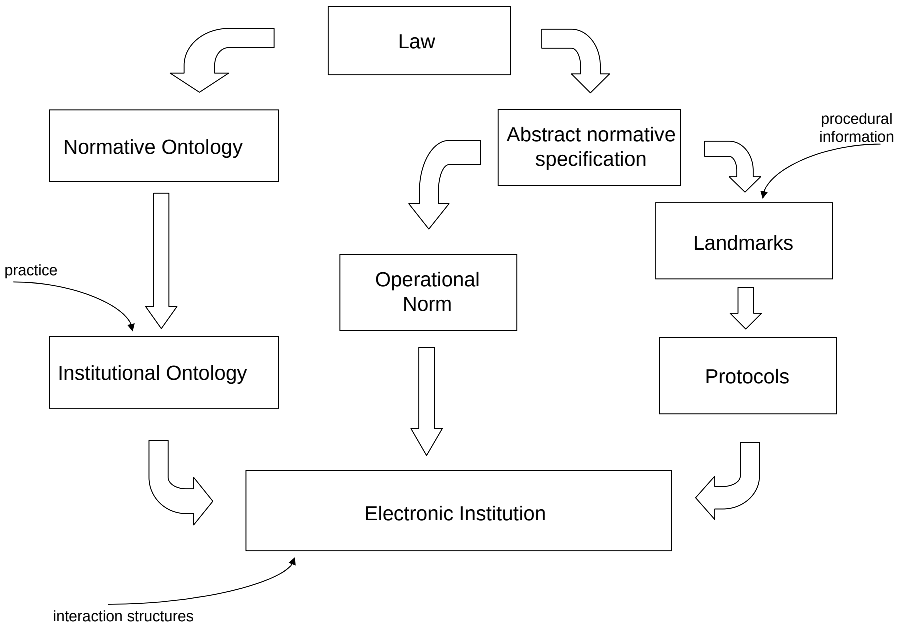


Figure 3.1: *From laws to electronic institutions.*

from the norms to create a prototypical protocol on a high level of abstraction (we call these important steps derived from the norms *landmarks*, and the structure that expresses the ordering over the landmarks a *landmark pattern*); 2) by using procedural information and the expected capabilities of the agents that will participate in the institution a protocol is created to give the agents a default manner for achieving certain goals in the institution.

The important elements that result from the methodological approach to the implementation of institutions derived from laws, as shown in figure 3.1, that characterise the institution are as follows.

- A common **ontology**, defining the meaning of concepts, the roles used in the institution and the relations between different contexts.
- A **normative specification** of the allowed interactions in the institution. This specification consists of *substantive* and *regimented* norms (a discussion about these types of norms is given below).
- **Protocols** to specify conventions in procedure mechanisms, giving a typical interaction profile which should work in any circumstance.
- An active **enforcement mechanism** to make sure that the participants of the institutions adhere to the normative specification.

The ontology is needed to specify how the agents interact, defining the communicative propositions that are used, and defining the roles and role hierarchy that will be used throughout the norms. The normative specification is the basis of the institution, specifying the legal and illegal actions in the environment. Denoted in a formal language, this specification can be used to derive the last two elements of the framework. The protocols define standard manners in which the legal interactions can take place in the institution. They provide a means for non-norm-aware agents to perform their tasks in the institution, or provide a guideline for norm-aware agents to follow (to show how things can be done, though are not necessarily the only way to do it, and can be deviated from if need arises). The norm enforcement is necessary to guarantee the safety of the system. Since we do not restrict the agent to only perform the allowed actions (i.e., we not only use regimented norms, but also substantive norms, thus expressing how it should be, while allowing violations of this 'ideal' view), it is required to check and enforce the proper ways of acting upon the agents in the institution. Much like in the real-world, instead of equipping all cars with speed-limiting devices, one specifies that speeding is illegal, and checks whether everyone in the institution adheres to that norm (even if one would opt for the regimented option of installing speed-limitation devices in cars, one would still have to check that no one tampers with the device and violates the norm).

In the following we look at the different steps in the methodology of figure 3.1, and explain how the elements that characterise the institution are obtained from the law.

3.2 Formalising Norms

The primary element of the framework is the normative specification of interactions. This specification follows from the laws and regulations that govern the domain of the institution, and is represented in some formal language. Norms, as used by institutions, specify what is allowed (legal) and prohibited (illegal) at a high level of abstraction and with a great deal of ambiguity. This abstractiveness and ambiguity of norms allows for a general specification of wanted and unwanted interactions in the environment without the need to consider each and every possible situation that might arise. By providing highly abstracted rules that regulate many different situations one gains a stable representation of the wanted interactions even for situations that were not considered at the time the norm was specified.

To use the laws and regulations that define the domain, a representation of these laws has to be given in a (formal) language, that can be used later on to implement the norms in the agent system. We already presented (classical) representations of norms in subsection 2.1.1. The representations discussed there focussed mainly on those based on the initial (modal logical) representation by [von Wright, 1951], describing normatives in terms of *obligation*, *permission* and *prohibition.* Another approach of representing norms, introduced in 1958 by Anderson [Anderson, 1958], reduces the deontic modalities to an alethic modal logic, it does not have a deontic content with the exception of a single propositional element V indicating the occurrence of a violation (or indicating the liability to some form of sanction). In these systems, the meaning of obligations, permissions and prohibitions is then given as a relation with respect to this special element (e.g., obligation to p is formalised as the necessity of not p leading to V). This manner of representing deontics in logic was adopted and enhanced in earlier mentioned formalisations such as those based on temporal logics, see for instance [Gabbay, 1976; van Eck, 1982], or dynamic logics, e.g., as done in [Meyer, 1988](a good overview of the developments in deontic logic can be found in [Meyer and Wieringa, 1993]).

The choice of a given formalisation of the norms depends on the aspects of the norms that are important for the design of the institution. Naturally, the deontic notions expressed in the norms should be captured by the formalism, but a formalism only able to capture deontics will lose a lot of details expressed in the norms, e.g., about the temporal orderings of states, relations between actions and action results, roles and role-dependent normatives, etcetera. The formalism chosen for representing the norms must have enough expressiveness for representing all these aspects from the norms that are needed for the design. Although some translations from one formalism to another is possible, like modelling actions (indirectly) in temporal logics or modelling time (relative to actions) in a dynamic logic, the choice of a formalism is a *ontological commitment* to the manner of expressing the norms (i.e., norms expressed in a dynamic logic will always intrinsically have an action-related flavour to them, and if one is more interested in the temporal relations expressed in the norms than in the action related aspects of the norms, it is best to choose a temporal based deontic logic instead [Grossi, 2004]). In chapter 4.2.1 we introduce a deontic representation of norms as an Anderson's reduction based in linear-time temporal logic, with added expressiveness for notions as actions and roles.

We mentioned before a difference in norms, being either *substantive* or *regimented.* The former type of norms are those that express the ideal and wanted situations in the environment, and when formalised in a deontic formalism, by means of obligations, permissions and prohibitions. These norms are to be adhered to by all the participants of the institution, and the institution has to see to it that this compliance is satisfied; hence, the need of an active norm enforcement mechanism (more about norm enforcement in section 3.4.2). The latter type of norms that we distinguish are those norms that will be implemented by means of constraints to directly regulate the possible actions that agents can perform and states that agents can reach. In the specification of the institution by means of these types of norms, one can distinguish the important actions/states that should never occur (expressed in regimented norms), and the actions/states that should ideally (not) occur (expressed in the substantive norms). The important difference between these two is that violations of the substantive norms, while unwanted, are possible, while compliance to the regimented norms is ensured at all times (because their compliance is inherent to the structure and working of the implementation).

In the remainder of this thesis we focus mainly on the substantive norms, and, moreover, assume that the implementation of a normative institution will consist of more substantive norms than regimented norms, since this is beneficial to the autonomy of the agents in the system, as will be explained in chapter 4. The substantive and regimented norms extracted from law will later on be used to design the institution (by translating the norms to implementable norms and creating an enforcement mechanism, and by extracting the important characteristics of the norms and using them to design protocols that can be used in the institution).

Like most of other elements of the framework, the norm specification can be taken, or derived, directly from the laws and regulations describing the domain (the relations of the other elements to the laws of the domain can be seen in figure 3.1). Looking at laws from human institutions we can observe three different types of norms (the following examples are taken from the Dutch Regulation on access to Police Registers).

- **Norms defining (refining) the meaning of abstract terms** (e.g., *"The criminal register administrator can be the Regional Police Force Commander, the Dutch National Police Force commander, the Royal Military Police Commander, the College of the Procurator-General or an official appointed by The Minister of Justice"*).
- **Norms defining (refining) an abstract action by means of subactions (a plan), a procedure or a protocol** (e.g., *"A request for examination [of personal data] [...] is sustainable after receipt of the payment of EUR 4,50 on account [...] of the force mentioning 'privacy request'"*).
- **Norms defining obligations/permissions/prohibitions** (e.g., *"Information should be removed from a register when it is no longer necessary for the purpose of the register."*).

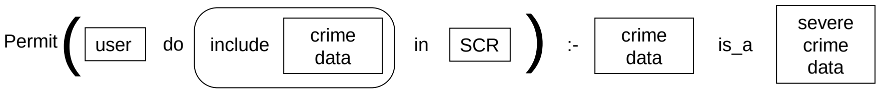
The first two types of norms specify elements that are important for the ontology of the institution. The first norm gives a meaning to (some of the) concepts used in the laws/regulations of the other types, while the second type of norms specifies the further meaning of (sub-)procedures that are mentioned throughout the laws/regulations. The third type of norms specifies the possible interactions that can take place in the institution, by specifying what is forbidden (illegal) and what is permitted (legal) in the institution and giving the responsibilities of the participants of the institution. Those norms are to be formalised to form the normative specification of the institution, which will be used to design the enforcement mechanism and the protocols used by the institution.

Let us look at the formalisation by means of some example norms from the Dutch law on Police Registers (from [Dutch DPA, 2006]).

Article 1

1. In this law and every depending clause it is meant by:
   - k. severe criminality register: a police register that is built for the execution of the police task, as far as it concerns:
     - 1°. crimes as described in article 67, first item, Code of Criminal Procedure, that have been devised or committed in an organised affiliation and, considering their nature or coherence to other crimes that have been devised or committed in organised affiliation, generate a serious impact on the legal order; or
     - 2°. crimes for which an imprisonment of eight or more years is dictated according to the legal definition;
     - 3°. crimes, constituted by general regulation of control, as described in article 67, first item, Code of Criminal Procedure, which are, considering their nature or connection to other crimes committed by the involved person, generating a serious impact on the legal order.

This first article is a norm that defines the meaning of a concept used in other norms in the law. It defines the meaning of the concept *severe criminality register* (SCR) in terms of what such a register is, and how it is used. This kind of information will be included in the ontology of the institution (we come back at this in the next section). Moreover, by dictating the meaning of a severe criminality register, this norm also expresses what can be (or should be) included in a severe criminality register, namely data concerning severe criminality (and the definition of what is considered a severe crime is give in 1° - 3°). Introducing in the ontology this meaning of 'severe crime data', we can model this implicit norm as[1]:



Basically, next to the ontological definition of what a severe criminality register is, this norm expresses the general condition under which information can be included in a severe criminality register. The intuitive notation used above was used as an intermediate language to represent norms needed for the ANITA project, because, at this point, we do not want to commit ourselves to the choice of a formal

[1]The representation below is not in a formal language, such as the deontic temporal logic we introduced in subsection 4.2.1, but more an intuitive representation of the concepts that are important for making a formalisation in such a formal language. The :- operator used, denotes a condition that must be fulfilled in order for the norm (expressed on the left-hand side of the operator) to hold.

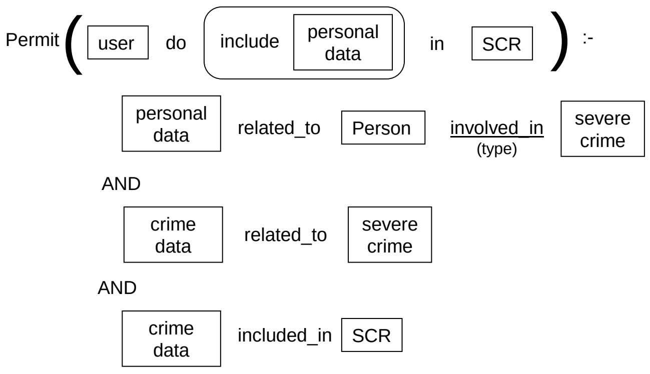
logic for representing the norms. This intuitive representation is the basis of the logical representation introduced in chapter 4.2.1 and will only be used in this chapter for illustrative purposes. In the other chapters of this thesis we will use the logical formalism that we introduce in chapter 4.2.1.

Another article in the law further elaborates what sorts of information can be included in severe criminality registers.

Article 13a

1. The inclusion of personal details in a severe criminality register occurs only when it concerns:
   a. suspects of crimes, for which the register is contrived;
   b. persons, against whom exists a reasonable cause to suspect that they are involved in the devising or committing of the crimes mentioned under a;
   c. persons who are related in some (specific) way to those mentioned under a and b.

Given the fact that a severe criminality register contains information concerning severe crimes, the norm expressed in this article extends the basic permission to include crime data (from article 1.1k) with the permission to also include personal data in severe criminality registers. This personal data, however, must be about people that are either a suspect of the crime, people that are suspected to be involved, or people that are somehow related to them (relatives, friends, contacts, etcetera). The modelling of, for example, 13a.1 item a then becomes.



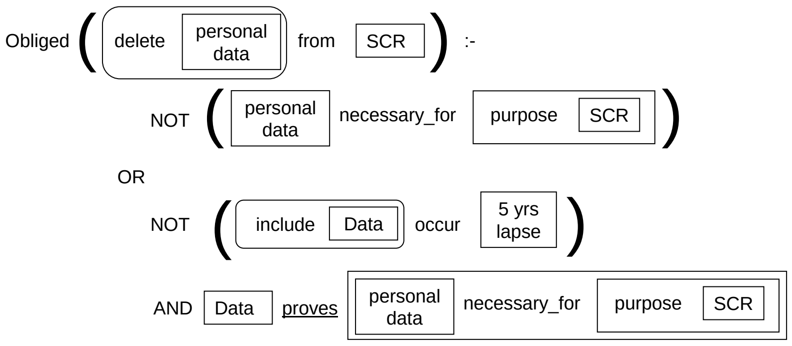
Similar modelings of the other elements of article 13a.1 can be given as well.

Next to these definitions of what information can be included in the register, the law also states when information in the register can be shared (and with whom), or when information must be removed from the register.

Article 13a

8. Information as mentioned in the first item is removed from the severe criminality register, and subsequently destroyed, if it is no longer necessary for the purpose of that register or else after a five year lapse from the date of the last entry/retrieval of information that proves the necessity of the registration of the person involved in accordance to the purpose of the register.

This article, which expresses that personal information needs to be removed from the register if it is no longer needed or when the information has not been used for 5 years, can be modelled as follows.



The set of formalised norms that result from this modelling process is what will be used later on in subsection 3.4.2 and subsection 3.5 to design the protocols and the enforcement to see to it that these norms are adhered to in the institution.

3.2.1 Combining Norms

Up to now we have focussed on the structure of 'individual' norms (i.e., one norm at a time), however, when considering a set of norms (e.g., norms in a regulation for a particular topic), we also have to handle the combination of these norms. Very often regulations start with general norms followed by exceptions to these norms. It means that formally the set of norms will be inconsistent when formalised as a set of first order formulas. In the theory of norms this issue is often related to the defeasibility property present in many AI applications [Pollock, 1987; Verheij, 1996].

A set of norms is called *defeasible* when a norm that can be logically derived from this set is no longer valid when an extra norm is added to the set of norms. In our setting we distinguish two following kinds of defeasibility.

- *Defeasibility of classification*: The semantic meaning of the concepts appearing in the norms may be extended, reduced or altered by the introduction of an extra set of norms.

- *Defeasibility of norms*: The impact and/or applicability of the obligations, permissions or prohibitions expressed in a given norm may be altered or even become inapplicable by the introduction of an extra set of norms introducing variations for some specific cases.

Although both types of defeasibility can be reduced to defeasible logics, their practical impact is different and therefore we treat these cases in different ways when implementing the combination of norms.

Defeasibility of classification

The defeasibility of classification is the most difficult case of defeasibility, as it involves the semantics of the concepts in the norm expressions. In human laws norms are expressed in a way that is open to interpretation, in order to extend their application to different particular contexts. This situation, highly useful for the stability and flexibility of human laws, poses an obstacle to the use of norms in computer systems, where a clear interpretation should be given.

For an example of defeasibility in the concept-level of norms, let us look again at the following excerpt of article 13a from the Dutch law on access to Police Registers.

Article 13a

1. The inclusion of personal details in a severe criminality register occurs only when it concerns:
   a. suspects of crimes, for which the register is contrived;
   b. persons, against whom exists a reasonable cause to suspect that they are involved in the devising or committing of the crimes mentioned under a;
   c. persons who are related in some (specific) way to those mentioned under a and b.

The law dictates that we can include personal details about several different sorts of persons in a police register. It does not, however, specify what personal details are. The concept is kept vague.

The Regulation on access to Police Registers, containing more concrete versions of the norms from the law, tells us what we can interpret as personal details[2].

Article 6

2. Concerning the persons, mentioned in article 5, under c, the source and the method of obtaining the information ought to be included. Furthermore, at most the following kinds of data can be included:

[2]Article 5 from the Regulation is similar to Article 13a.1 in the Law, although a bit more concrete.

a. the Municipal Basic Administration number, the last name, forename(s), address, place and date of birth, gender;
b. financial information;
c. information about the nationality;

⋮

In the third item of article 6 of the Regulation on access to Police Registers we see the defeasibility of the classification of the concept *personal details.*

Article 6

3. If the category of persons meant in second item concerns a CIE-informant then the informant-number, the informant-code and reference to the informants register, as mentioned in article 5 of the Regulation, are included at the most.

The third item of article 6 shows that when the related person (as mentioned in article 13a.1c) is actually an informant of the Criminal Investigation Unit (in Dutch: Criminele Inlichtingen Eenheid, CIE), the classification of personal detail changes. Instead of being allowed to include the information as mentioned by article 6.2, we can, in this case, only include the information mentioned in article 6.3. Thus, in the case of a CIE-Informant, the information about the nationality, for example, does not classify as personal detail (anymore).

In the case of implementation of norms with defeasible concepts, some kind of procedure or automated decision-making process should be created in order to classify a certain situation in terms of the defeasible concepts that appear in the regulation (e.g., a procedure to decide if, in a specific case, which subset of information can be considered *personal information*). If it is not possible to automate such decision-making for all possible cases (because, for instance, such a decision is highly context-dependent and therefore lots of expertise on a give field is needed), then the decision should be delegated to a human expert (e.g., the definition of what is a *reasonable cause to suspect*).

Defeasibility of norms

Norms in human regulations are formulated in a manner that is very similar to non-monotonic logics and default reasoning techniques [Antoniou, 1997; Pollock, 1987; Verheij, 1996]. That is, laws are generally specified in several levels of abstraction. On the most abstract level, normally the constitutional laws, a law defines the *default*, i.e. it defines what actions to take (or which predicates should hold) when certain conditions hold or specified situations arise. The 'lower' levels of abstraction (e.g. applied law and decrees) generally specify exceptions to this default. They specify that certain situations do not follow the general norm and ask for a different approach.

Article 13

1. Any procurement that occurs directly through automated manner is recorded, as far as these procurements are not dispensed by decree of the Minister of Justice.

5. A procurement is not recorded in accordance with the first item, when it is a result of a linkage and a report of the linkage has been drawn up.

Article 13.1 specifies the obligation to record in the system log files any automated procurement of data that has not been stated in a decree from the Minister of Justice. This describes a quite clear situation, easy to be included in the decision making of the recording procedure of the system. We can express article 13.1 as follows[3]:

$$\begin{array}{ll} A13.1 & \mathsf{OBLIGED}((system, \mathsf{DO}\, record(procurement_i, sys\_logs)) \\ & \mathsf{IF\ NOT}\ (origin(procurement_i, decree(Minister\_Of\_Justice)))) \end{array}$$

The addition of Article 13.5 suddenly *defeats* what is stated in Article 13.1, as it introduces a special, exceptional case where the first article does not hold. In principle we can express Article 13.5 as follows:

$$\begin{array}{ll} A13.5 & \mathsf{NOT}\ (\mathsf{OBLIGED}((system, \mathsf{DO}\, record(procurement_i, sys\_logs)) \\ & \mathsf{IF}\ (origin(procurement_i, linkage_j)\ \mathsf{AND}\ reported(linkage_j, sys\_logs)))) \end{array}$$

By this example we can see how defeasibility impacts in the reasoning process. There will be situations where both norms A13.1 and A13.5 will be triggered, and therefore two contradictory results (the obligation of recording and NOT the obligation of recording) appear. In this simple example it is quite clear that A13.5 overrides what is stated in A13.1 (by considering A13.1 the *default case* and A13.5 an *exceptional case*), but solving collisions at run-time for all possible combinations of norms is a complex and time-inefficient task.

Introducing the handling of defeasibility of norm sets in the reasoning mechanism is not a good option, as there is no efficient implementation of defeasible logics (such as default logic). Therefore there is a need to bypass defeasible reasoning, by solving all collisions off-line. Depending on the rate of changes in the law, there are two possible options to handle defeasibility of norms in implementation.

- *Changes in the law almost never occur*: As defeasible reasoning is computationally too complex, one possible option would be to avoid the defeasibility directly in the logical representation of the norms (that is, the logical representation extracted from the human regulations re-structures the conditions for the base case and the exceptions in a way that it is not defeasible). In

[3]Formal semantics of operators similar to those used in the examples below will be given in section 4.2.1 and beyond. For now, the intuitive reading of the operators will suffice.

order to do so, the conditions that express when the exceptions occur should be introduced in the original norm as preconditions. For the previous example, expressions A13.1 and A13.5 can be merged in a single, non-defeasible expression as follows:

$$\begin{array}{ll} A13.1\_5 & \mathsf{OBLIGED}((system, \mathsf{DO}\, record(procurement_i, sys\_logs)) \\ & \mathsf{IF}\,(\mathsf{NOT}\,(origin(procurement_i, decree(Minister\_Of\_Justice))) \\ & \ \mathsf{AND\,NOT}\,(origin(procurement_i, linkage_j) \\ & \ \mathsf{AND}\, reported(linkage_j, sys\_logs)))) \end{array}$$

The problems of this approach are that a) defeasibility should be completely handled by the designer or the knowledge engineer while building the computational representation, and b) there is no longer a direct mapping from each of the articles of the human law to the norm expressions in the computational representation, and therefore maintenance of the computational representation when there are changes in the law becomes highly difficult (e.g., what is to be changed in expression A13.1_5 if there is a new article that expresses an exception to the exception in Article 13.5?).

- *Changes in the law often occur (periodically)*: In this case the alternative is to build a defeasible computational representation of the norms, where each of the articles in the human law is mapped. In order to use the computational representation, an automated process searches for those norms that become defeasible because of other norms and solves the problem by moving and/or adding conditions. The original defeasible representation of norms should include new objects in the object language to express the relations between expressions [Prakken, 1997; Prakken and Sartor, 1997b]. For instance, Articles 13.1 and 13.5 could be represented as follows:

$$\begin{array}{ll} A13.1 & \mathsf{OBLIGED}((system, \mathsf{DO}\, record(procurement_i, sys\_logs)) \\ & \mathsf{IF}\,(\mathsf{NOT}\,(origin(procurement_i, decree(Minister\_Of\_Justice))) \\ & \ \mathsf{AND\,NOT}\,(\mathsf{CONDITIONAL\_EXCEPTION}(A13.1)))) \end{array}$$

$$\begin{array}{ll} A13.5 & \mathsf{CONDITIONAL\_EXCEPTION}(A13.1) \\ & \mathsf{IF}\,(origin(procurement_i, linkage_j) \\ & \ \mathsf{AND}\, reported(linkage_j, sys\_logs))) \end{array}$$

  In this case the representation explicitly specifies that expression A13.5 only impacts the conditions in expression A13.1. This information will be used by the automated process to generate the final, non-defeasible representation, getting automatically the expression A13.1_5 above.

The advantage of this second approach (proposed in [Vázquez-Salceda et al., 2005], and based on earlier work presented in, e.g., [Prakken, 1997; Prakken and Sartor, 1997b]) is that each time there is a change in the law, the change can be easily made in the defeasible computational representation, which then automatically can be processed to eliminate defeasibility before its use. This way of implementing defeasibility resembles very much the idea of circumscription put forward by McCarthy, see [McCarthy, 1980]. Although after this seminal work has been published both criticisms and improvements have been given (see, e.g., [Lifschitz, 1994]) it seems that in our particular setting we can suffice with the original idea of circumscription. One of the main reasons is that we do not encounter situations with several contradictory exceptions to a norm which require more complicated treatment. This will occur with agents that have to deal with norms coming from different origins, but not for institutions. The norms that institutions have to incorporate coming from outside the institution can be prioritised easily on the basis of their origin and conflicts of exceptions are not present within the context of one institution.

3.3 Defining the Ontology

The ontology of a system is a shared *conceptualisation* of the concepts, terms and entities that exist in the system, and the relations that hold between them. It is an abstract simplified view of the world that is represented [Gruber, 1993; Bench-Capon and Coenen, 1992]. The objects and relations between objects defined in the ontology give a model of the domain of the system, but also form the basis of the vocabulary of the agents participating. The predicates used by the agents to communicate to one another (e.g., FIPA performatives) are a part of the ontology. This basic function of the ontology, i.e., defining the meaning of the concepts and giving the agents the means to communicate, make the ontology a vital part of an agent environment. The ontology, as an explicit specification of the conceptualisation of the domain, in a sense, a representation of domain knowledge that is needed to make the system work, can be (partially) derived from the laws that describe the domain.

As we have shown earlier in figure 3.1, the basis of the ontology is taken from the law. We have shown in section 3.2 that there are three types of norms in law; norms that define the meaning of abstract concepts, norms that define the meaning of abstract actions, and norms that express the expected behaviour in the institution. The first two types of norms are the ontological definitions, included in the law itself, of concepts in the law. The concepts and meaning expressed in these norms can directly be used in the building of the ontology of the institution. Consider again, for example, article 1.1k of the Dutch law on Police Registers, which is an example of such a norm.

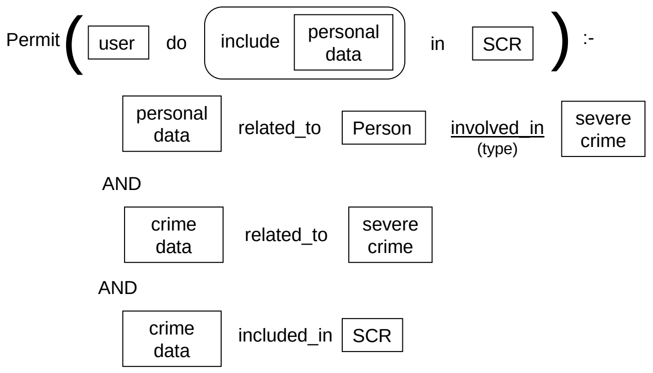
Article 1

1. In this law and every depending clause it is meant by:
    k. severe criminality register: a police register that is built for the execution of the police task, as far as it concerns:

1°. [crimes as described in the Code of Criminal Procedure devised or committed in an organised affiliation]; or
2°. crimes for which an imprisonment of eight or more years is dictated according to the legal definition;
3°. [crimes as described in the Code of Criminal Procedure, constituted by general regulation of control].

This norm defines the meaning (on an abstract level, still) of several concepts. Clearly it gives a meaning to what a severe criminality register is, but, indirectly, also defines what is considered as a severe crime. Additionally, it introduces new concepts that are not given a meaning, like 'the police task', or 'general regulation of control'. These concepts need to be added to the ontology as well, and are either used as grounding concepts or need to be given a meaning.

Moreover, during the formalisation process of the norms of the third type, as described in section 3.2, new concepts and actions will be used in the formalisation of the norms as well. Let us look again at the informal representation of article 13a.1:



Although we have a meaning of the concepts 'SCR' and 'personal data' (given by article 1.1k of the law and article 6 of the Regulations, respectively), this representation uses concepts such as 'criminal data' or 'person', which need to be given a meaning as well. Some of these concepts can easily be linked to concepts already in the ontology (e.g., 'criminal data' clearly defines 'data' about a 'crime', given that these latter concepts are already in the ontology), but in some cases it is needed to consult a domain expert for making these connections. The concepts and the relations between these concepts, as derived from law, can be used to make a relational dependence graph, as shown in figure 3.2.

Due to the extensive use of abstract and vague concepts in norms, to give the norms a flexible and adaptable nature (a norm does not describe a single situation, but can be applied to a range of similar situations), many abstract concepts exist in the ontology that have no apparent meaning in a concrete environment (such as the implementation of an institution). A connection, or translation,

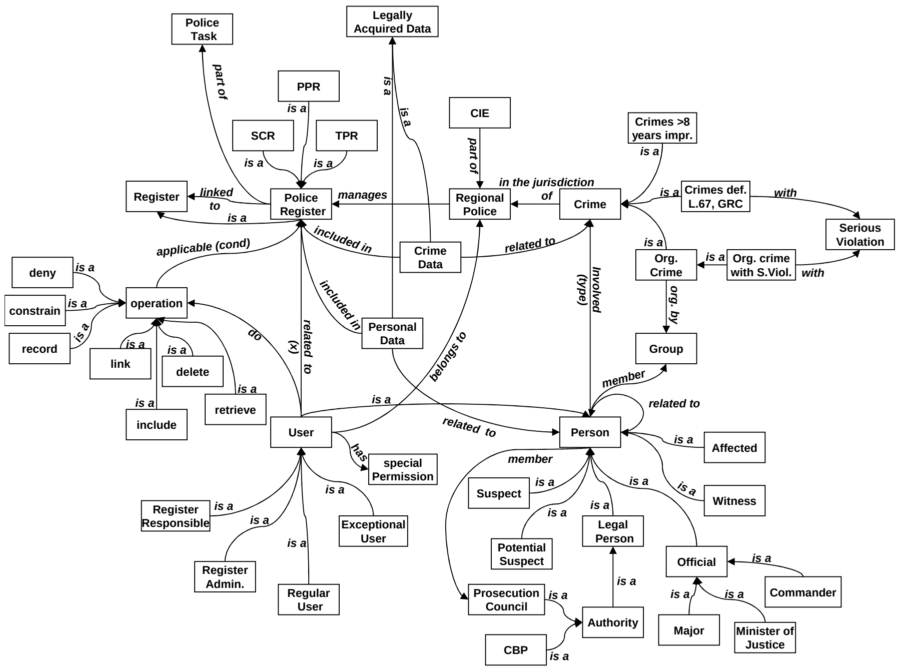


Figure 3.2: *Concepts and relations taken from the Dutch law on police registers.*

between these abstract concepts and concrete concepts is needed. Part of this translation consists of linking the norms to the concrete situations that can appear (we get back to this in subsection 3.4.1), but a concrete meaning must be given as well. This means that connections have to be made between these abstract and concrete concepts (the concrete concepts are introduced into the ontology by looking at the practice and using (predefined) interaction structures that will appear in the implementation of the institution). This connection, or mapping, between abstract and concrete concepts can be done with a *counts-as* operator, such as used in [Grossi et al., 2006b]. Although such an operator holds many intriguing and challenging aspects, these topics are beyond the scope of this thesis. We assume that such connections between the abstract and concrete concepts can be made (e.g., by defining conceptual subset relations between the abstract and concrete concepts), but refrain from going into details about formal semantics and implementations of this operator (readers interested in these aspects of counts-as should check, for instance, [Grossi et al., 2004, 2006b]).

The counts-as connections between the abstract concepts used in the norms and the concrete concepts used in the practice require that the concrete concepts are a part of the ontology. These concrete concepts are taken from elements that do not (directly) follow from the law that govern the domain, such as, for instance, interaction structures used in the implementation or domain knowledge taken from practice (e.g., 'police registers are managed by a (regional) police force', or 'the criminal investigation unit (CIE) is a part of the police force'). The abstract concepts and their relations to each other, combined (through the counts-as) with the concrete concepts and their relations taken from the practice make the ontology that will be used for the implementation of the institution (the abstract concepts define the meaning of the normative specification, the concrete concepts are used for the vocabulary of the agents and the meaning of the concrete situations that arise, and the counts-as provides the connection between these two to allow norm enforcement and norm reasoning).

3.4 From Normative Specifications to Institutions

The translation from laws and regulations in natural language to a formal representation (the abstract normative specification) is only the first step of the process of implementing the norms. Since laws and regulations are expressed at a high level of abstraction, to allow the law to cover a wide variety of situations and to be used for an extensive period of time without the need for modification, it is hard to link them to the concrete situations that arise in the practice. To make the normative specification useful in an agent institution, an interpretation of the norms is needed, which should only contain concrete (institutional) meanings of the vague and abstract terms used in the norm and which might contain procedural information that can be used to simplify the enforcement of the norm. This process of interpreting the norms to make them useable for a single context, i.e., the institution, is called *contextualisation.*

Although norms that result from the contextualisation process are concrete and contain only concepts that are meaningful in the institution, these norms need another translation before they can be implemented. Norms, especially those expressed in deontic logic, only have a declarative meaning, i.e., how things should be, while abstracting from operational meanings, i.e., how it should be achieved. Moreover, there is more than one way to enforce a single norm and procedural information (which is not part of the norm) will have to be used to decide how the norm is best implemented. This second translation process of adding additional operational and procedural information to the norms is called *operationalisation.*

3.4.1 Contextualising Norms

The contextualisation process is meant to give a link between the abstract terms and concepts used in the abstract normative specification and the concrete situations and concepts that exist in the institutional practice. Where norms contain terms such as 'fair' and talk about actions like 'discriminating', these concepts have no clear meaning in the institution itself. There are, however, states and (sequences of) action(s) in the institution that can be classified as an interpretation of one of these vague concepts in the *context of the institution.* These concrete interpretations are highly context dependent and can differ from institution to institution. For example, whether something should be considered to be *personal data* depends on the context in which the concept is used; what counts as personal data in a hospital might not count as personal data in a police register. The problem then remains how these connections between the abstract and concrete concepts are made in order to create concrete norms that are useable in the institution.

The connection between abstract concepts in the norms and the concrete concepts used in the institution are defined in the institutional ontology (or over the abstract and concrete ontologies of the institution, if both exist). By means of a *counts-as* operator, see [Grossi et al., 2004, 2006b], it is given that a certain concrete concept can be considered an instantiation of an abstract concept used in the norms. Consider, for instance, the following example norm of an auction house, expressing the obligation to identify oneself upon entering an auction:

$$\mathsf{OBLIGED}((participant\,\mathsf{DO}\,identify)\,\mathsf{IF}\,(participant\,\mathsf{DO}\,enter(auction)))$$

The action $identify$ in this norm has an abstract meaning and can be implemented in various different manners. To implement this norm the meaning of this abstract action must be defined, which is done by connecting the abstract action to concrete action(s) through the use of a *counts-as* operator:

$$\begin{array}{l}[participant\,\mathsf{DO}\,give(certificate, manager)\,\mathsf{AND}\\ manager\,\mathsf{DO}\,check(certificate)]\,\textsc{counts-as}\,participant\,\mathsf{DO}\,identify\end{array}$$

describing that giving an identification certificate to the auction manager, and the manager checking this certificate is seen as an implementation of the $identify$ action. These counts-as definitions, defining the scope (and applicability) of the abstract concept, are highly context dependent and not necessarily one-one definitions.

Implementing these counts-as definitions is achieved by extending the existing ontology of the institution, which already consists of all the abstract concepts used in the norms of the institution, with the concrete concepts that are used in the practice and the relation between the abstract and concrete concepts. This relation is defined as a conceptual subset relation, specifying that the ontological meaning of the concrete concepts is included in the ontological meaning of the abstract concepts. In the case of our example, this would mean that the ontological meaning of the actions $give(certificate, manager)$ and $check(certificate)$ are included in the meaning of the abstract action $identify$.

Using these counts-as definitions we can bring the abstract norms to a concrete level, where we assume that:

$$\text{If } C\,\textsc{counts-as}\,A \quad \Rightarrow \quad \mathsf{OBLIGED}(A) \text{ means } \mathsf{OBLIGED}(C)$$

That is, the obligation to the abstract concept A is rewritten as an obligation to the concrete concept C (given that C is the concrete concept interpreting the abstract concept A). But more importantly;

$$\text{If } C \text{ COUNTS-AS } A \quad \Rightarrow \quad \text{violating } \mathsf{OBLIGED}(C) \text{ means violating } \mathsf{OBLIGED}(A)$$

If the concrete norm is violated (i.e., the norm containing C), it can be considered as a violation of the abstract norm (i.e., the norm containing A), and thus the measurements devised for enforcing the abstract norm are applicable to violations of the concrete norm.

After interpreting the abstract concepts of the norm, we obtain a norm that is only appropriate for the chosen domain, but, on the other hand, only contains concrete concepts that are easily linked to states/actions in the implementation. In some cases, though, trying to detect a violation would involve making lots of checks, which could be computationally hard or totally infeasible from the institution's point of view. For instance, checking whether every participant of the institution is able to identify oneself at any given time can be very hard, particularly in very crowded institutions.

Moreover, there might be norms in the institution which would have a severe impact on the institution if the norm would be violated. As recovery from such violations (normally done by sanctions and repairs) would be (nearly) impossible, it would be wise to try to reduce such violations to an absolute minimum. For example, as murder is, in itself, a very severe violation of the norms of a society, any measures that can be taken to limit the occurrence of this violation should be taken (e.g., the prohibition of owning (fire-)arms).

In both cases, one is trying to simplify the enforcement process such that it either becomes feasible to detect the violation, or protect the system from very harmful violations. This process of contextualising norms can be done in two ways. Either the norm is translated to smaller and simpler norms which are easier to check but ensure the compliance of the original norm, or the norm is translated to a set of constraints that ensure the compliance.

As mentioned earlier, we differentiate between *substantive* and *regimented* norms, where substantive norms express what is desired while enforced by checking for violations and punishing violating agents, and regimented norms are enforced by means of direct constraints on the agents behaviour (for instance, by making certain resources or actions unavailable for certain roles in the institution). The latter form of norms is used for those norms which express situations that absolutely should not happen, since their violations are either too severe or unrecoverable for the institution. Implementing a norm as a substantive norm, however, is not always just checking whether the ideal situation expressed in the norm exists, and reacting to deviations of this ideal situation, since such checks can be very resource expensive or even unfeasible to make. Substantive norms can express situations that are hard to check (for instance, checking the obligation to have an identification certificate at all times is very time consuming), or the violations of the norm can be very hard to react to (for instance, punishing agents for disconnecting in the middle of a negotiation is hard since it is unknown whether the agent will return to the institution). To solve these problems, other norms (both substantive and regimented) can be used to allow the checking of compliance of the norm or to simplify the enforcement of the norm. The newly introduced norms will have to be simpler to verify (they should be easily checkable for violations, and easy to enforce) and should introduce the means to make the violations of the original norm more transparent and allow a better and easier application of sanctions to the violations of the original norm.

In a sense, the contextualisation of norms to implementable versions of these norms is in accordance to the following heuristics.

1. Norms with violations that are very severe (i.e., system breaking) and (nearly) unrecoverable should be regimented (access to the actions/resources that trigger the violations should be denied to all agents in the system).
2. Norms which are hard to check should be implemented by introducing (substantive or regimented) norms that allow the checks to be made more easily or ensure the compliance more easily; e.g., implementing a norm covering a task t by giving a procedure for completing t and expressing that the procedure should be used every time t must be done.
3. Norms which express violations that are hard to sanction should be implemented by the introduction of (substantive or regimented) norms that allow sanctions to be carried out; e.g., the obligation to give certain information upon entering that makes the agent traceable, paying security deposits, giving account or credit card information and/or signing an agreement that states that any damage done to the institution or other participants can automatically be subtracted from the account or credit card.

Let us explain this by means of an example. Consider an institution that implements an auction house, where agents can join and bid on items they want or need. In such an environment a norm holds stating that when an agents bids on an item it has to pay for the item if it won the auction.

$$\mathsf{OBLIGED}((buyer\ \mathsf{DO}\ pay(Price, seller))\ \mathsf{IF}\ done(buyer, won(Item, Price)))$$

Violations of this norm occur, for instance, because an agent does not have enough money to pay, the agent does not want the item anymore or the agent simply disconnects (unintentionally or on purpose). Although the violation of this norm can be detected easily, sanctioning the agent and repairing the situation might be difficult (especially if the agent disconnects). To avoid these situations, one can choose to implement this norm by means of a constraint; upon entering the institution all agents have to deposit an amount of money (for security) that they will get back when leaving the institution if no violations have occurred:

$$\mathsf{OBLIGED}((agent\ \mathsf{DO}\ pay(Security\_Fee))\ \mathsf{IF}\ done(agent, enter(Institution)))$$

However, if a violation of the mentioned norm occurs, this money can be used to pay for the items, thereby sanctioning the agent. This means that our original norm has been implemented by introducing a norm that is easier to enforce (i.e., agents are obliged to pay security before entering and are not allowed inside the institution without paying), which generates the constraint (or mechanism) that is used for enforcing the original norm. Thus, instead of implementing one norm which was hard to enforce, we have implemented two norms (which were extracted from the original norm) that are easily enforced.

3.4.2 Operationalising Norms

Implementing normative systems based on the elements presented above requires an active form of norm enforcement, i.e., the institution (or institutional agents) should check at all times whether the norms of the institution are obeyed and punish those agents that do not comply with the norms. This enforcement of norms is needed for a few reasons: 1) being an open multiagent system, where agents that are not designed by the implementer of the institution can join at any time, no guarantees can be given about the inner workings of the participants of the institution. To expect all agents to function with an internal norm-model (may it be hard-coded into the agent's goals, or through normative reasoning capabilities) is much unwarranted. 2) Fully regulating the interactions between participants through some sort of intermediate level between the agents and the institution (as currently done in the AMELI implementation of ISLANDER institutions, [Esteva et al., 2004a]) requires the designer of the institution to fully lay out the patterns of interactions possible (designing the interaction protocols), thereby either minimising the possible interactions to a minimal set of all interactions allowed by the norms or increasing the complexity of the development significantly (we get back to these points in subsection 4.1.2). Moreover, fully regimenting interactions between participants to the procedures thought of by the designer severely limits the most important aspect of agents (at least, in our view, the most interesting aspect), namely autonomy.

Though implementing norms by fully regimenting them has its disadvantages, the opposite, implementing them by means of norm enforcement, has its problems too. The work on norm representations, and accordingly the norm specification that we mentioned above and present in 4.2.1, has been focused on the *declarative* aspect of the norm. While this aspect of norms is important, telling what is legal and illegal, giving the ideal situation, investigating the expressiveness of norms and verifying the consistency of a given set of norms, it only describes what should be, but lacks a description of how it should be achieved or what should be done to make sure the norms are not violated. For the implementation of norm enforcement, an operational meaning of the norms is required, describing exactly when the norms are violated, what causes these violations, and what the responses from the institution to such violations should be. Trying to implement the norms without this operational information is very hard.

The idea would be to translate the norms, obtained from laws and regulations, to checks that can be made in real-time by specifically designed institutional agents (agents responsible for the enforcement of the norms, called *Guardian Agents* in [Fox and Das, 1999], or *Police Agents* in [Vázquez-Salceda, 2004]) to see if norms have been violated. To be able to make these checks (in an efficient manner) the norms need to be given an operational meaning, i.e., which actions and situations in the implemented institution cause violations of the norms and what should be done when a violation occurs. This process of *operationalising* norms consists of adding information to the declarative representation of the norm (which expresses, in logic, what is right and wrong and can be used by agents capable of normative reasoning) to express: 1) a clear and checkable representation of when the norm is violated, 2) a representation of the mechanisms best used to detect these violations, and 3) the actions that are needed to be performed after a violation has occurred to recover from the unwanted state (including punishing the violating agent and reversing, or rather fixing, any ill-effects that might have been caused).

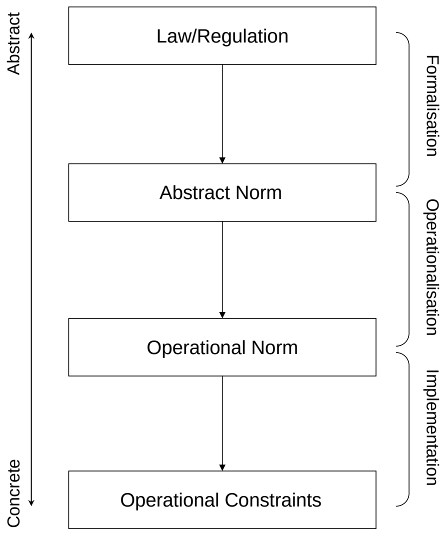


Figure 3.3: *Deriving norm enforcement mechanisms from law.*

The process of implementing the norm enforcement, from top (the law) to bottom (the institution, as shown in the middle section figure 3.1) is then as depicted by figure 3.3. First the laws and regulations are translated to a formal representation (as mentioned in 3.2). This formal representation expresses, on a high level of abstraction, what should and should not be, and is translated, through contextualisation and operationalisation, to an operational norm, that not only expresses what should (not) be, but also expresses what should be done to see to it that the norm is adhered to. That is, the abstract norm derived from the law is annotated with concrete operational information that is needed for the enforcement of the norm. Lastly, the operational norm is translated to constraints that can be checked by the enforcers in real time, and that express what actions the enforcers should take when a violation of the norm is detected. This process of implementing norms by active norm enforcement is discussed in detail in chapter 4.

3.5 Designing Protocols

Specification and implementation of institutions based on the elements presented above appear to provide a safety and security that the environment needs to ensure trust among the agents participating in the institution. The resulting institution is, however, only useable for agents that can understand and reason about the normative specification to decide what the best course of action is to achieve their goals. Agents without these norm-reasoning capabilities or agents that do not understand the normative specification of the domain will not be able to (efficiently) work in the institution. For these kinds of agents guidelines must be given to inform them of standard ways to achieve their goals. These guidelines, or protocols, express the best course of action for a given situation while making sure that none of the norms are violated. Next to being essential for non-norm-aware agents, these protocols are also useful for agents that are able to reason about the norms, since the protocols specify the default actions for a given goal (the norm-aware agents can, however, choose to deviate from the protocol under certain circumstances).

While it is possible that protocols or procedures exist in the real-life institutions governed by the laws and regulations, these protocols need not be complete or might even be unfeasible to use in an agent institution. While one can start with the protocols and procedures that govern a domain, and base the interaction structures of the agent institution on these real-life protocols (such as done in, for example, implementations of the fish market [Noriega, 1997], and as the general approach of creating an ISLANDER institution [Esteva, 2003]), this can go very wrong in highly-regulated domains, since more than often the connections between the procedures used in real-life and the norms that govern the domain are lost, and translating the procedure to an agent implementation might introduce steps that violate the normative specification that holds for the domain (in these circumstances it is necessary to verify whether the implemented procedure is norm-compliant).

The introduction of new protocols, if no real-life procedures exist, or changing the existing protocols is very hard, due to the discrepancy between the norms and the procedures. Norms are expressed in abstract terms and are, therefore, hard to be directly translated to the concrete actions of which the protocols consist. To solve this problem we introduce an intermediate level between norms and protocols, an idea inspired by the relation between laws and practice, where regulations are used to give more concrete and procedural meaning to the abstract concepts used in the norms. The idea is to extract the most important characteristics from the norm, which express what should be done, and in which order this should be done. We call these characteristics *landmarks*, and the structure which denotes the order in which the landmarks need to be obtained the *landmark pattern*. It is not hard to see that this landmark pattern, given that it includes all the important states and the order in which these states need to occur as expressed by the norms, is in fact a prototypical protocol on a high level of abstraction.

Similar to the relation between law and practice, where procedural information is added to the normative specification of law by means the regulations, we add procedural information (information that helps increase the efficiency and feasibility of the prototypical protocol) to the landmark pattern. This information is obtained from the practice and is used to strengthen the landmark pattern (in a sense, we add landmarks and landmark orderings to the landmark structure to express this additional information in the landmark pattern).

The final step is translating this strengthened landmark pattern, that contains the states that are dictated by law and the states that were added for efficiency and feasibility reasons, to a protocol that can be used in the institution. This translation is made by combining the states expressed in the landmark pattern to the (expected) capabilities of the agents in the system. In chapter 5 we go into detail about this process, and show how the translations from norms to landmark pattern, the strengthening of the pattern, and the translation from landmark pattern to protocol are done in a highly-regulated domain.

3.6 Discussion

Designing and implementing institutions from law is a difficult process. While research has been done on how to express and reason about norms and how to implement electronic institutions, not much work has been done on the connection between these two; how to create an electronic institution from law.

On the one hand, much research, in the field of legal theory as well as computer science, has been done on norms, but this research has been mainly focused on the representation of norms, reasoning about norms, and on the analysis of the expressiveness of these formal representations. Little to no research has been done, however, on the operational meanings of norms and how norms should be implemented in concrete systems, creating a gap between the normative specifications and the concrete agent systems.

On the other hand, some research has been done on agent societies, where the interactions between agents are controlled by the environment (or the agent platform); e.g., electronic institutions and electronic organisations. A concrete real-life organisation is translated to an agent implementation, taking the procedures and concrete rules of the organisation as a starting point, and building the agent system bottom up. Although very successful for environments that are (heavily) procedure-driven (e.g., auctions), this approach does not work for the highly-regulated domains that we are interested in, where the interactions between the agents is specified by (a large amount of) abstract norms. Instead of starting at the interaction structures given by the real-life examples (which usually lack a clear connection to the norms of the domain), we need to start at the norms to define the expected behaviour and create the interaction structures to match this.

In this chapter we have shown a methodology to built institutions from a specification given by laws and regulation. While this is not a complete methodology, it makes the first steps that are needed and tries to cover the most important aspects of institution design. Unfortunately, there is not much research available on this topic; the most important knowledge about how this task is done best is held by the people who design legal systems and work on legislation and the creation of laws, but that knowledge is, unfortunately, too abstract to be of any use to the concrete solutions that we seek.

In the beginning of this thesis we set out to formalise and implement the legal domain governing the information exchange between police registers (as part of the ANITA project), which has been the main motivation behind the creation of the methodology presented in this chapter. The problem of exchanging information in this context was assumed to be hard because of differences in the application of rules by each police district. The idea was to create an electronic institution based on the norms specified by the law, that controlled the information exchange between the different registers of the police districts (each represented by an agent), where the registers themselves had the autonomy to enforce local rules on the information exchange, as long as they adhered to the general norms extracted from the law that were enforced by the institution. However, when it became apparent that the law was about to change, solving the disparity between local and general norms, we had to abandon the idea of an implementation that would be used in the practice (which was the goal of ANITA), and had to switch to a more suitable domain. In the remainder of this thesis, although ANITA examples will be used here and there, we will focus more on the domain of organ transplantation, because it suits our purpose better.

The methodology we presented in this chapter creates a framework for institutions on the basis of the law. This framework consists of an ontology for the institution, taken partially from the law and partially from the practice; a normative specification derived from the law; and protocols and norm enforcement, both created on the basis of the normative specification, to implement the institution. In the following chapters we look at some of these different aspects in more detail; how norm enforcement is created from an abstract specification of the law (chapter 4), and how protocols can be created from a formal specification of the law (chapter 5).

Chapter 4

Implementing Normative Institutions

In previous chapters we have looked at the relation between norms (derived from laws and regulations) and agent societies, in particular agent-mediated institutions. Agent-mediated institutions, or electronic institutions, are open agent societies (open multiagent systems, see chapter 2) where heterogeneous agents can join to complete tasks. This heterogeneity of the agents that join the systems makes it increasingly difficult to ensure a safe and reliable environment for the agents to work in. To solve this problem, as we discussed in chapters 2 and 3, norms are used to specify the behaviour expected from the agents. These norms, derived from human laws and regulations, however, are specified on a high level of abstraction, making it very difficult to work with in the concrete situations that arise in these institutions.

Naturally, the correct behaviour of agents in a system does not automatically follow from the specification of the rules of behaviour. The norms need to be implemented in either the agents or the system itself to guarantee the compliance to the norms and thereby creating the safety and reliability that is expected. There are several design choices when it comes to implementing norms; for instance, whether the norms are implemented in the agents or in the institution, or whether violations of the norms are allowed, etcetera. We take a look at these design choices in section 4.1. In the remainder of the chapter we set out to give the procedure of implementing the norms from an institutional perspective, as proposed in chapter 3.

This process of implementing norms from an institutional perspective, which needs to overcome this gap between the abstract specification of the norm and the concrete constraints that are needed in the implementation, consists of several steps as shown in figure 4.1 (already presented in chapter 3). First the *laws* and *regulations*, expressed in natural language, have to be formalised in a (deontic) logic, to capture the declarative meaning of the law/regulation, expressing what

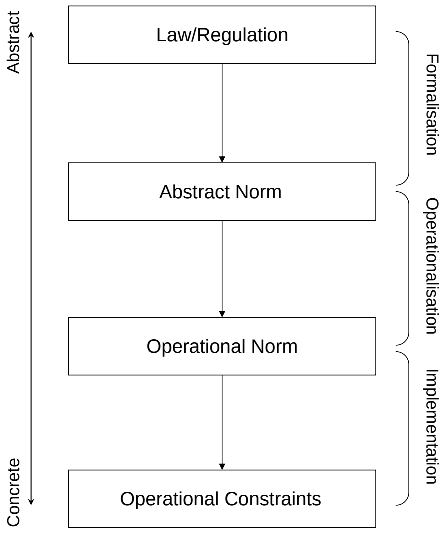


Figure 4.1: *From law to concrete implementation.*

is right and wrong in the institution. The *abstract norm*, expressed in deontic logic, resulting from the formalisation, should then be contextualised and operationalised to give the norm operational meaning, expressing how the norm should be implemented; e.g., how does one check for violations of the norms, what should be done after a violation has been detected. The *concrete norm* resulting from this process of adding the operational information to the deontic representation is expressed in a norm frame, which represents all those elements of the operationalised abstract norm that are needed for the implementation. The final step of the implementation consists of translating the elements of the norm frame to *operational constraints* that help the institution in enforcing the norm.

In subsection 4.2.1 we present the representation formalism for the abstract norms, and extend this formalism to the norm frame needed for the representation of concrete norms in subsection 4.2.2. In this section we also discuss the relations between the formalisms and how the step from abstract norms to concrete norms is made. Later, in section 4.3 we introduce the operational constraints by which we implement the norm frame to obtain the normative institutions. In this section we also explain how the norms from the norm frame are translated to these constraints.

4.1 Different Approaches to the Implementation of Norms

Norms, as described in chapter 2, are used in normative systems to regulate and control heterogeneous agents by describing what is right and wrong behaviour, thereby defining the expected and ideal behaviour of the agents participating in the system. The implementation of such norms, thus trying to make the agents behave in compliance with the norms, is, however, not always specified and can be achieved in different manners. This implementation should consider a) how the agents' behaviour is affected by the norms, and b) how the institution should ensure the compliance to the norms. The former is related to the *implementation of norms from the agents' perspective*, by analysing the impact of norms on the agents' reasoning cycle. The latter is related to the *implementation of norms from an institutional perspective*, by implementing a safe environment (including the enforcing mechanisms).

Furthermore, in the case of norms implemented from an institutional perspective, a choice can be made on the operationalisation of the norm enforcement. In general the enforcement of norms comes down to either:

- Defining constraints on unwanted behaviour; or
- Detecting violations and reacting to these violations.

The former method relies on the introduction of *artifacts* to help control (regiment) the agents such that unwanted behaviour cannot occur at all in the institution. The latter is more flexible by monitoring the behaviour of the agents and punishing them when they behave in a manner that is not expected/wanted (as defined by the norms).

In the following we will look at some of the possibilities that can be used to achieve norm compliance in normative agent systems.

4.1.1 Agent Perspective

The agents' perspective on the implementation of norms is about the impact that norms have on the behaviour of agents in the normative systems. The norms pose restrictions on the agents which require them to adapt their behaviour in order to complete their goals. This means that the agents will reason about the sequences of actions best used in order to obtain a specified goal, or even, drop a goal as it is unreachable, because of normative restrictions. This behavioural perspective on norms can be implemented in two manners.

Norm-Controlled Agents

The oldest and, arguably, simplest manner of making sure that the agents in the institution perform in accordance with the norms, is designing the agents to behave exactly as expected by the norms, thereby 'hard-coding' the norms into the agents desires, intentions and capabilities. The agents are not (necessarily) aware of the norms, but were, at design, instructed to behave in the specified manner. Such agents are norm-compliant by default, and cannot perform any actions that would violate the norms; such choices are not incorporated into their programming.

This approach does, however, have the disadvantage that all the agents in the institution must be designed and implemented by the developer of the institution, as agents that have not been designed by the system developer cannot be trusted (one cannot be sure that 'external' agents have the same, rigorous programming that allows them to be norm-compliant). Another disadvantage of this approach is the reduction of the level of autonomy on the part of the agents, as the agents in the system are fully programmed to do what is required, thereby not leaving much room of variations or autonomous decisions that might be helpful to the system.

Norm-Aware Agents

A more intriguing, and more difficult approach to the implementation of norms from an agent's perspective is making the agents able to reason about the norms and the consequences of their actions in a normative domain [López y Lopez, 1997; van Kralingen, 1995; Visser, 1995; Verhagen, 2000]. This means that the agents themselves have the capabilities to reason about when certain actions might lead to violations of the norms, and reasoning about the results of a violation in order to determine whether violating the norm (and thus accepting the sanction on the violation) has a higher gain for them than behaving according to the norm.

Norm-aware agents are interesting from two different perspectives. Firstly, such agents, when used in highly-regulated domains, could be able to perform better than agents that just follow a given protocol. Since protocols in a highly-regulated system are (or at least should be) designed to be norm-compliant in all possible situations, they might prescribe inefficient procedures given the current situation. Agents that are capable of reasoning about the norms, however, can in every different situation, decide for themselves on the best possible course of action. Moreover norm-aware agents can, because they are more adaptive than agents that do not reason about the norms, drop their goals sooner if they derive that it is no longer feasible (due to normative restrictions) to attain that goal. Generally speaking, norm-aware agents are more flexible than agents that are not able to reason about norms, as agents that reason about the norms are not necessarily restricted to protocols that were defined on forehand.

An example of a framework based on norm-aware agents is described by López y Lopez in [López y Lopez, 1997; López y Lopez et al., 2001; López y Lopez and Luck, 2002]. This framework defines normative agents as those agents whose behaviour is shaped by the obligations it must comply with and prohibitions that limit the kind of goals that it can pursue. To this extent they define a norm frame which includes *addressees* (the agents to whom the norm applies), *beneficiaries* (those who the addressees are focused on), *normative goals* (the goal that is to be achieved or avoided as specified by the norm), *contexts* (the states of the environment when the norm is active), *exceptions* (states when the norm is not active) and *rewards* and *punishments* (responses to the compliance or violation of a norm). This norm frame (describing what is supposed to be achieved/avoided in which context) is then linked to the actions by means of a relation between a specific action and a norm. Either an action *benefits* a norm (making it possible to be compliant to the norm when the action is executed) or it *hinders* a norm (executing the norm makes it impossible to be norm-compliant). It is then stated that, for a normative agent, all actions benefiting the norms are permitted and all actions that hinder the norms are prohibited.

The other interesting perspective to norm-aware agents is that they can be employed to enforce norms on other (norm-aware) agents. This basically means that the norm-aware agents, while trying to be norm-compliant themselves, are also empowered to enforce the norms on other (non norm-aware) agents. The capability to reason about norms for one's own actions can be extended with the capabilities to reason about how norms (should) influence the behaviour of other agents. In this manner the norm-aware agents can be used as 'enforcers' of the norms on other agents, punishing them when those agents violate a norm, and trying to uphold the norms in the system. This approach is closely related to the enforcement from an institutional perspective (the norm-aware agents become part of the institution, and act on behalf of the institution), which will be discussed below.

4.1.2 Institutional Perspective

The norm implementations mentioned above are good for making agents perform better in normative domains. Since the agents know the rules and can reason about them, they can adapt their behaviour easier. Moreover, the agents might be even able to reason about the 'cost-benefit' of violating certain norms of the domain, making them able to reach goals in manners unforeseen.

However, these implementations do not necessarily ensure the compliance of the norms; hard-coded agents can violate the norms, for instance because of faulty programming, or worse, because unexpected situations arise that lead to norm-breaking behaviour by the agent. Similarly, norm-aware agents, that are able to reason about the norms and know the penalty for breaking a norm, are not guaranteed to work violation-free. As we discussed, these agents might actually *choose* to violate a norm, accepting the sanction as the price for reaching a certain goal in a certain manner. Even then, we are not taking into account agents that are either not designed with the norms of the system in mind, or are trying to violate the norms on purpose, which can just as easily join the normative system.

The main point here is that someone or something should be dealing with illegal behaviour of the agents, regardless of whether they are aware or ignorant of the norms. This leads to the conclusion that some measures must exist in the society to regulate and enforce the norms on the agent in that society. The norms regulating the agents are part of an institution, and it is this institution's responsibility to ensure that the institution is safe and reliable for the agents that are joining it. If, for instance, in an automated auction house no measures to control cheating and illegal bidding procedures exist, no rational agent (or its designer) would want to participate in that auction house. Moreover, since some of these electronic institutions are an extension of the real world, and therefore governed by external norms, this safety and reliability of the system can be demanded by these external norms. E.g. the electronic institution discussed in [Vázquez-Salceda et al., 2002; Vázquez-Salceda, 2004] used for the allocation of organs (for the use of transplantation), is governed by norms from, among others, the Spanish Government and European Union. A system built for that purpose, governed by those norms, is required to be safe and reliable because of this relation to the real world. If the norm compliance of the agents in the system cannot be guaranteed (thereby failing to guarantee the norm compliance of the system as a whole), the system would not be accepted by the governing agencies.

This, of course, means that agents that are violating the norms are, very much, a problem of the institution itself and should be dealt with by the institution. In the following we will look at the implementations of norms from an institutional perspective.

Regimentation of Norms

As mentioned in the beginning of this section, one of the manners of implementing norms in an institution is through the constraining of the behaviour of agents. If one does not want agents to perform a certain action a in the institution, simply '*disable*' that action a; if agents are not supposed to have access to certain resources, make sure those resources are not available to them.

In contrast to regimentation in real world institutions, controlling agents (which are software entities) can be done more simply. Agents access the electronic institution through platforms and services of the platform that are provided by the designer of the institution. If certain behaviour of agents is unwanted, the access to specific services on the platform can be reduced or even refused.

In its most extreme form the regimentation totally defines what agents can and cannot do. This means that the institution provides the agents with a protocol that takes the agents step-by-step through all the procedures in the institution (prescribing all actions from the moment the agents enters the institution up to the moment the agent leaves the institution). Deviations from the protocol are impossible and, at any time, no actions other than the ones specified in the protocol can be performed (e.g., if in a certain situation the protocol allows the agent to do either a or b, the agent can only choose to do either a or b, doing c is not allowed and will not be accepted by the institution).

This heavily restricting method of norm implementation has been used in the ISLANDER [Noriega, 1997; Rodriguez, 2001] and AMELI formalisms [Esteva et al., 2004a; Esteva, 2003]. The functionality of an institution in those formalisms is expressed in terms of a *dialogical framework* (which is sort of the definition of the communicative actions possible between the agents) and a *performative framework*. The latter is a protocol-like definition of the scenes in the institution and the order in which these scenes are to be played out. Scenes themselves are, again, kinds of protocols describing the possible (allowed) interactions between the agents in the institution. Entities called *governors* [Noriega, 1997; Esteva, 2003; Esteva et al., 2004a] are used to ensure that agents joining the electronic institution comply with these protocols. A governor, as mentioned in [Noriega, 1997; Rodriguez, 2001; Esteva, 2003], is an agent in the institution that acts on behalf of an agent joining the institution, thereby providing a (social) interface to the institution. Governors are programmed to only follow the protocol specified in the institution, and work as a filter for the illocutions (interactions) of agents joining the institution; governors only pass those illocutions to the other agents that comply with the protocol, thus guaranteeing protocol compliance for all agents in the institution. E.g. in the simple example given above, if an agent tries to do c in the mentioned situation, the governor ignores this and waits for the agent to do either a or b.

The governors in [Noriega, 1997; Rodriguez, 2001; Esteva, 2003] are envisioned as agents filtering messages of external agents, possibly extendable to even advise agents that are joining the system on the best possible actions in a given situation, helping them to work efficiently in the institution. However, in the current implementations, cf. [Esteva et al., 2004a], governors are not as 'multi-talented' as presented here, and can only filter the messages of the agents joining the institution; only passing norm-compliant messages and disregarding the rest. Because of this limited use of concept of governors, these governors can not really be seen as agents but show a striking resemblance to the coordination artifacts presented in [Omicini et al., 2004]. A coordination artifact is an 'embodied entity' specialised to provide a coordination service in a MAS, quoted from [Omicini et al., 2004]:

> "Coordination artifacts are infrastructure abstractions meant to improve coordination activities automation; they can be considered then as basic building blocks for creating effective shared collaborative working environments, alleviating the coordination burden for involved agents".

This is exactly what governors do in the current implementation of [Esteva et al., 2004a], they are abstractions that help on the coordination between agents and the platform, making sure that the agents follow the designed protocols and procedures of the institution.

The main advantages of the full regimentation approach of implementing norms lies in its total control over the agents. All the norm enforcement, and the compliance to the norms, are handled at the design of the system, thus reducing the number of norm-breaking faults in the system to those made by the designer himself at the design and implementation of the system. If designed and implemented correctly, however, these systems are certain to ensure the norm compliance of the agents. Unfortunately, this also leads to a major disadvantage of the approach, since the translation from vague and abstract norms to the concrete protocols used in the system can be very hard, making the design of such a (good working) system very difficult.

Another major disadvantage of fully regimented systems is the level of autonomy of the agents in the system (or more precisely, the lack thereof). Since the agents are only allowed to perform the steps prescribed by the protocols, only having to make minor choices, their autonomy, one of the important and characteristic aspects of an agent, is severely restricted. This means that the advantages of having autonomous agents participating in the system, which can react to unexpected situations and function in circumstances that were not thought of at the design of the system (but are similar to those circumstances that the agent was designed for), disappear or are at least greatly reduced. Some of this reactiveness and flexibility can be captured in making a more complex protocol (the protocol takes more possible situations in consideration), but this, however, only enlarges the complexity of the design of the system, thus increasing the former mentioned disadvantage.

Using Violations and Sanctions

Instead of regimenting all the agents in the system, thereby creating a rigid institution with agents that are very restricted in their autonomy, a more flexible approach can be obtained by using the other method of implementing norm enforcement. As we mentioned in the beginning of this section, norm enforcement can be implemented by either defining constraints on unwanted behaviour (as we did in the regimentation approach), or by detecting violations of norms and reacting to these violations. Contrary to the regimentation where unwanted actions simply cannot be done, in this approach it is possible to execute unwanted actions (illegal actions or actions that lead to illegal situations); however, doing such an action incurs a response, namely a punishment for doing that action.

Implementing a more flexible enforcement mechanism for institutions requires that checks are being made by the institution to determine whether the norms that hold are violated. If a violation is detected, an action must be taken by the institution to punish the violating agent. This, however, does not mean the institution itself must become a being capable of checking and punishing agents, as, like in human institutions, the enforcement of the norms can be delegated to a specific set of agents (i.e., enforcing or police agents, cf. [Vázquez-Salceda et al., 2004, 2005]). This selective group of enforcing agents can contain agents that are designed by the developer of the institution (we call such agents *internal agents*), or agents that are designed by someone else (to keep the analogy, we call these *external agents*). Since these agents are empowered with the responsibility of keeping the order in the institution and trying to ensure the safety and stability of the system, giving such powers to agents that one does not know well (i.e., agents that are designed by others) can be dangerous and requires a lot of trust in the working of the external agent and the developer of that agent.

Using violations and sanctions for implementing norm enforcement allows for a much higher level of autonomy of the agents, which now have the freedom to choose which action is going to be done and are not restricted to a fixed and pre-determined protocol, thereby allowing the agents to solve situations that were not considered during the design of the institution. Moreover, since following the norms becomes a choice, the meaning of being norm-compliant actually has gained a value in this kind of system. In a system where all the norms are implemented by using regimentation, norm compliancy is not a choice (actually not being norm-compliant is not even possible), thus being more a feature of the system instead of being an obtained quality of the agents in the system. This quality of the agents, in non-regimented systems, can be used, for instance, as the basis of trust and reputation measures used in the institution. For instance, in an electronic auction house implemented using violations and sanctions, where it is the norm that one should pay for the goods one has bid on and won, agents that are known to comply to this norm can have a higher standing than agents that are known to break this norm (no-one wants to do business with agents that do not pay for the goods).

Allowing agents to violate the norms, which is normally considered to be a bad thing (system designers rather not have agents breaking the rules), can actually be more effective in certain cases. If the procedure that the agents have to follow becomes much more expensive, it might be better to allow for some small violations. For instance, consider a mail delivery protocol, where no violations can occur. This would mean that upon delivery the receiver of a package will have to check the contents of the package to determine that the package was delivered in perfect condition. Although this would eliminate the possibility of a violation, it is very cost-ineffective, since the delivering agent will have to wait for the checking of the package. It would, of course, be even worse if the mailing of a package took place through a number of mailing services, where every transaction (from one mail-service to the next) would require a lot of time to check the condition of the package. If, instead, the delivering agent is allowed to violate the norm, thus allowing the possibility that a package would be delivered that is not in perfect conditions, the delivering procedure becomes much faster and cost-efficient (the delivering agent no longer has to wait for the checking procedure and can deliver more packages in the time that it would take in the case of the other delivering procedure). Of course, now that violations can occur, mechanisms have to be made to detect violations and punish the violator(s) (receivers will need to have to possibility to file a complaint, and investigations to determine the responsible party will have to be made), but this second protocol is, in the most cases, still more efficient.

In order to verify whether a norm is violated by agents in the system, checks need to be made to determine whether the illegal action has actually been done by the agent, whether the action was actually illegal at the time it was done, et cetera.

In certain circumstances these checks can be either infeasible to perform (for example, checking whether an agent believes that something is true) or are very time- or resource-expensive. Sometimes this can be improved by the introduction of resources or mechanisms to facilitate these checks. In other cases it might be easier to use regimentation for norms which would otherwise require checks that are utterly infeasible.

Although using violations and sanctions to enforce norms on agents in the institution is good for the level of the autonomy of the agents, there might be norms whose violation would pose a too severe impact on the system. Violating norms that break the entire system, that are not easily (or impossibly) recovered, are good examples of norms that one would not want to enforce by means of violation detection and sanctions. For enforcement of these kinds of norms the regimentation approach described above should be used instead.

4.1.3 Discussion

In the previous sections we have discussed norm implementations from an agent perspective as well as from an institutional perspective. As we already mentioned, the agent perspective is for making agents aware of the rules and regulations in the normative system, allowing them to increase their efficiency in such environments. This increased efficiency is possible since the agents can reason about the 'cost-benefit' of doing actions that (might) violate the norms, thereby being able to reach goals that might seem unreachable.

Implementations from an agent perspective are, however, not enough to guarantee the compliance to the norms, as no guarantee can be given of the agents joining the system; the agents joining might not be aware of the norms (or capable of reasoning about them), or might not be willing to follow the norms of the system (agents that are made with the sole purpose of disrupting the system are very much a possibility). This leads us to conclude that some measures must be taken to regulate and control the agents in the system. As the norms are a part of the institution, it is exactly the institution that should ensure that the institution is safe and reliable, which means that it is the institution's responsibility to enforce the norms on all agents joining that institution, which leads to the conclusion that if safety and reliability of the institution are the issue, it is not the agent perspective on implementing norms, but the institutional perspective on norms that is important.

The regimentation approach of implementing norms from an institutional perspective and the violation detection and handling approach are the two extremes of the scale of what is possible and reasonable for implementing normative institutions. We already mentioned that the former approach, restricting the agents in such manners that the only thing doable in the institution is that which is defined on forehand, thus making the agent only follow a protocol, leads to a rigid institution where the autonomy of agents is heavily restricted. As mentioned, such systems are safe (no norms will ever be violated), but can become very inefficient and unable to handle unexpected circumstances.

Implementing a system by merely checking and reacting to violations can ensure the needed increase in autonomy on behalf of the agents, allowing them to overcome these problems, and making it possible to work more efficient while being norm-compliant (most of the time). However, in those circumstances where the checks required to determine whether a violation has occurred are too hard or too resource-demanding, violations might go unnoticed, which is undesired. Moreover, there are some norms that would, if violated, pose a too severe impact to the system, so that violations of these norms are undesired.

Of course, many choices for combining these two mechanisms can be devised, to ensure the high (enough) level of autonomy for the agents, while being able to ensure that certain norms can never be violated (thus protecting the system from the severe consequences that such a violation would have). The choice between implementing a norm by a constraint or by detecting the violation is, next to the apparent reasons described above, dependent on the nature of the system and the resources available. In situations such as the mail delivery example described earlier, for instance, it might be more cost-effective for the institution to allow the violation of damaging the package by the delivering agent, and using a system to sanction the agent in the unfortunate case that such a violation occurs. However, if the domain would be the delivery of dangerous volatile materials, violations of the 'safe delivery' norm would be very undesirable, and a more restricting protocol might be used to constraint the involved agents such that no violations could ever occur, thus protecting the safety of the system.

It is important to notice that, if violation detection and sanctioning is chosen to implement a norm, the enforcement of the norm is expressed in terms of other norms, i.e., norms imposing checks and norms specifying the reactions to the occurrences of a given violation. As a matter of fact, as described in [Hart, 1961; Grossi et al., 2006a], one can isolate two types of norms involved in the specification of institutions; there is a set of *substantive norms* which consists of those norms which describe the behaviour desired by the institution, and there is a set of *enforcement norms* consisting of norms regulating checks and reactions on violations of other norms[1]. Consider again the example of the mail delivery, where the norm stating that the package should not be damaged during transportation is the substantive norm, while the enforcement norms consist of the obligation to investigate a complaint about a damaged package and the obligation to punish the deliverer (and the obligation to compensate the complaining agent).

Of course, the enforcement norms, which direct the agents enforcing the substantive norms on the 'normal' agents, can also be violated and would, if not regimented, require enforcement themselves (see figure 4.2). In principle, this process of creating new enforcement levels can be iterated *ad infinitum*, which

[1]Note that we already introduced the concept of *substantive* norms in chapter 3, in contrast to *regimented* norms, where substantive norms can be violated (the reactions to the violation of a substantive norm is specified by an *enforcement* norm, as mentioned by [Grossi et al., 2006a]) while regimented norms cannot be violated.

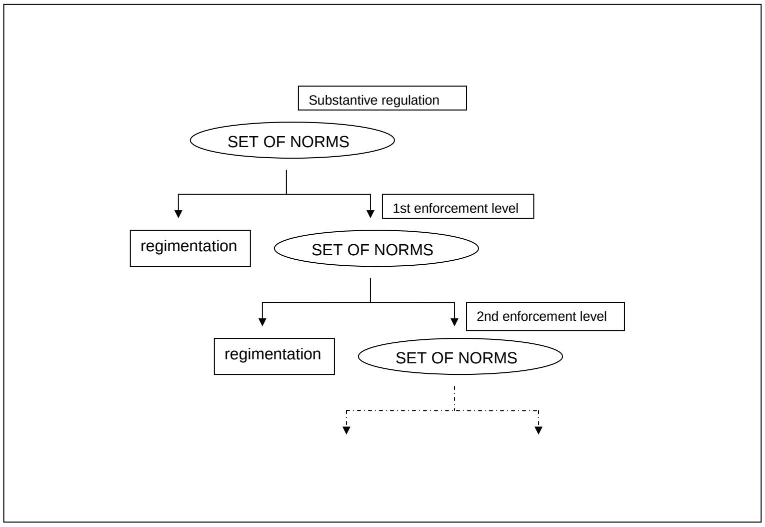


Figure 4.2: *Normative levels in institutions.*

would be very inefficient. There are, however, two alternatives to the infinite adding of normative levels to enforce the norms on the previous level; either all norms on the final layer are regimented, thereby discounting the need for another level, or by ignoring violations on the final level, which is exactly what is used in human institutions where the rulings of the supreme court are supposed to be final (even though they might be violating a norm).

We already established that the violations of substantive norms can have added effect for the institution, as it can increase the efficiency of certain protocols, and allows for more flexibility and autonomy of the agents. The question then remains how effective the allowance of violations on the (first) enforcement level are, and whether it will benefit the institution (or the design of the institution) to allow for the violation of enforcement or the addition of more than one enforcement level. In general, since violations are undesired[2] (and thus need to be acted upon), failure in detecting a violation or not reacting to a violation would seem very undesirable. Moreover, since the last level of enforcement is likely to be implemented by the developer of the system, it would be prudent to regiment the enforcement at the first level of enforcement, thus only allowing violations to occur at the substantive level. Only when exceptions to sanctioning (the enforcers

[2]Although we allow the possibility of violations occurring, it does not mean in any sense that violations would be come a desired or pleasurable event. Violations remain what they always were, undesired or illegal situations; A ‘flag’ to tell enforcers that things have gone wrong and a response is needed to restore safety in the institution.

can decide not to sanction a violating agent, e.g., because the violation was 'not that bad') are considered, ignoring the violations on the first enforcement level (or adding a second enforcement level) would be a reasonable choice.

4.2 Representing Norms

Before attempting to actually implement norms in one of the forementioned manners, the meaning of the norms has to be clarified by means of a formal representation. This representation has to make clear which actions and situations are accepted (legal) and which are not accepted (illegal). Such a representation of the norms allows us to capture the precise meaning of the norms and helps us to reason about the norms. This aspect of the norms is known as the *declarative aspect* of the norm, and has, traditionally, been captured in deontic formalism as mentioned in chapter 2. In subsection 4.2.1 we present a formal deontic logic based on linear-time temporal logic to express this declarative part of norms.

This, however, is not enough to implement the norms using the institutional perspective mentioned above, as additional information is needed to operationalise the norm. The declarative aspect of the norm tells us when something is wrong, but not what should be done to detect that something has gone wrong and, more importantly, what should be done when something has gone wrong. The latter information is called the *operational aspect* of the norm, and has to be attached to the declarative representation to make clear how the norm should be implemented. In general, the declarative part represents what legality and illegality mean in the system, given in terms of obligations, permissions and prohibitions. It describes *what* is allowed for *whom* and *when*, thus giving a description of the ideal. The operational aspect, on the other hand, defines what has to be done to prevent the violation of the norm and what should be done in case a norm has been violated. It expands the normative defining *how* the norms may/must be applied. It is exactly this latter part, the operational aspects of the norm, that is very important for the implementation of norm enforcement mechanisms. Therefore, as proposed in [Vázquez-Salceda et al., 2004, 2005], we represent the complete norm by annotating the (declarative) representation, discussed in subsection 4.2.1, with fields containing the operational information (presented in subsection 4.2.2), such that it contains all the information that is needed to implement the norms from an institutional perspective when using violations and sanctions. Of course, parts of these operational annotations are derived (or at least influenced) by the declarative representation (the detection of violations of a norm is, for instance, clearly quite dependent on the declarative representation). These relations between the operational and declarative aspects will be discussed later on when we look closely at the different parts of the norm frame in subsection 4.2.2.

The norm frame presented in the following is tailored to the institutional approach and contains all information needed for implementing norms through the definition of violations and sanctions in electronic institutions. The information in the norm frame does, however, also cover operational aspects of the other implementation methods described above. We discuss this relation to the other implementation approaches in subsection 4.2.3.

4.2.1 Declarative Representation of Norms

The expression of the declarative part of the representation (we call this part the *norm condition*) can be done in any kind of deontic formalism. In this work we use a linear-time temporal logic, based on temporal logic such as used in [Pnueli, 1977; Kröger, 1987], which we extend with deontic operators to express the obligations, permissions and prohibitions that are in the norms.

The overall representation needs to be readable by the agents in the system; the agents participating in the system need to be informed about the norms that hold in the system, thereby allowing them to plan the best (norm-compliant) path of action to accomplish their goals. Especially the agents that are able to reason about the norm benefit from this, as the explicit specification of the norms can be used by them to reason about and plan their goals in such a manner that they will achieve them in a norm-compliant way. To achieve this we use a machine-readable format for expressing the declarative aspect of norms.

Let $\mathscr{L}_P$ be a classical propositional logic, $\mathscr{A}$ be a set of atomic actions (including a `skip` action to express agents that are doing nothing) and Agents be the set of agents. The classical propositional logic $\mathscr{L}_P$ is constructed from atomic propositions, the universal contradiction (falsum, representing the classical $\bot$) and the logical operators NOT, AND, OR, $\mathsf{IMPLIES}$, EQUIV (representing the classical symbols $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$, respectively), which are used to construct formulas in the normal manner. We extend this base logic ($\mathscr{L}_P$) to a linear-time temporal logic ($\mathscr{L}_{TP}$) by including the following temporal operators with their intuitive meaning: $\mathsf{NEXT}\, A$ (A holds in the time point immediately after the reference point), $\mathsf{ALWAYS}\, A$ (A holds at all time points after the reference point), $\mathsf{SOMETIME}\, A$ (there is a time point after the reference point at which A holds), and two different *until* operators $A\, \mathsf{UNTIL}^*\, B$ (weak until; A holds at all following time points up to a possible time point at which B holds or holds indefinitely) and $A\, \mathsf{UNTIL}\, B$ (strong until; A holds at all following time points up to a, mandatory occurring, time point at which B holds). We include actions in the framework by defining $\mathsf{DO}_a\, \alpha$ (a does α next) and $\mathsf{DONE}_a\, \alpha$ (a did α some time ago) formulas using the models of $\mathscr{L}_{TP}$, which we explain later.

Formally, this means that the *temporal propositional logic* $\mathscr{L}_{TP}$ (with kernel $\mathscr{L}_P$) is an extension of the propositional logic $\mathscr{L}_P$, where the alphabet of $\mathscr{L}_{TP}$ is that of $\mathscr{L}_P$ with the addition of the operators NEXT, ALWAYS, $\mathsf{SOMETIME}$, UNTIL, DO_a, and DONE_a, which gives us the following definition of a formula in $\mathscr{L}_{TP}$.

Definition 4.1 (Formulas in $\mathscr{L}_{TP}$)

1. *Every atomic proposition is a formula.*

2. *if α is an action and a an agent then* $\mathsf{DO}_a\,\alpha$ *and* $\mathsf{DONE}_a\,\alpha$ *are formulas.*
3. *If A and B are formulas then* $\mathsf{NOT}\,A$, $(A\ \mathsf{AND}\ B)$, $\mathsf{NEXT}\,A$, $\mathsf{ALWAYS}\,A$ *and* $(A\ \mathsf{UNTIL}^*\ B)$ *are formulas.*

Next to the operators specified in definition 4.1 we derive, in the usual manner, the remaining operators as abbreviations of those specified: $A\ \mathsf{IMPLIES}\ B \equiv \mathsf{NOT}(A\ \mathsf{AND}\ \mathsf{NOT}\,B)$, $A\ \mathsf{OR}\ B \equiv \mathsf{NOT}(\mathsf{NOT}\,A\ \mathsf{AND}\ \mathsf{NOT}\,B)$, $A\ \mathsf{EQUIV}\ B \equiv (A\ \mathsf{IMPLIES}\ B)\mathsf{AND}(B\ \mathsf{IMPLIES}\ A)$, $\mathsf{SOMETIME}\ A \equiv \mathsf{NOT}\,\mathsf{ALWAYS}\,\mathsf{NOT}\,A$, $A\,\mathsf{UNTIL}\,B \equiv \mathsf{SOMETIME}\,B\,\mathsf{AND}\,(A\,\mathsf{UNTIL}^*\,B)$. We also introduce the universal tautology (truth, analogous to the classical propositional $\top$) as an abbreviation: $\mathsf{truth} \equiv \mathsf{NOT}\,\mathsf{falsum}$.

To give meaning to these formulas we define a possible worlds, Kripke-style, semantics, where the model consists of sets of Agents and Actions in the normative system together with a sequence of temporally related states defining the changes over time. States in this model give meaning to the atomic propositions in $\mathscr{L}_{TP}$ much as in classical propositional logic, defining what is true or false in the different time moments described by the model. Atomic actions, which are taking place between the defined moments in time, are related to the agent performing the action and the moment the action is performed. We define a model for $\mathscr{L}_{TP}$ as follows.

Definition 4.2 (Models of $\mathscr{L}_{TP}$)
a model $\mathcal{M} = \langle \mathsf{Agents}, \mathscr{A}, \mathbf{W}\rangle$ *for the propositional temporal logic* $\mathscr{L}_{TP}$ *consists of:*

- *a set of* Agents *operating within the system;*
- *a set of actions* $\mathscr{A}$ *that can be performed by the agents in the institution; and*
- *an infinite sequence* $\mathbf{W} = \langle \eta_1, \eta_2, \eta_3, \ldots\rangle$ *of states where every* $\eta_i = (v_i, \aleph_i)$ *with*
  - v_i *is the local valuation function for the propositions of* $\mathscr{L}_{TP}$*; and*
  - $\aleph_i : \mathsf{Agents} \to \mathscr{A}$ *is the function determining the action done by every agent in state i of the model.*

The models for $\mathscr{L}_{TP}$ consist of a linear sequence of states representing a possible execution trace of the system described by $\mathscr{L}_{TP}$, The states ($\mathbf{W}$) of this sequence defines the truth of propositions at the various time moments. Moreover, the states link every agent to an action, thus describing what each agent does every moment in time (meaning that every time moment every agent does exactly one action from $\mathscr{A}$, this also means that agents that are inactive for one or more states are represented by linking them to the `skip` action in those particular states). Using these models, the semantics of formulas of $\mathscr{L}_{TP}$ can then be defined as follows.

Definition 4.3 (Semantics of $\mathscr{L}_{TP}$)
For every model $\mathcal{M} = \langle \mathsf{Agents}, \mathscr{A}, \mathbf{W} \rangle$, formulas A and B, $a \in \mathsf{Agents}$, $\alpha \in \mathscr{A}$, and $s, t, u, v \in \mathbb{N}_0$:

$$
\begin{array}{rcll}
\mathcal{M}, s \vDash A & \Leftrightarrow & v_s(A) = \mathbf{true} & \textit{for every atomic proposition} \\
\mathcal{M}, s \vDash \mathsf{NOT}\, A & \Leftrightarrow & \mathcal{M}, s \nvDash A & \\
\mathcal{M}, s \vDash A \;\mathsf{AND}\; B & \Leftrightarrow & \mathcal{M}, s \vDash A \textit{ and } \mathcal{M}, s \vDash B & \\
\mathcal{M}, s \vDash \mathsf{DO}_a\, \alpha & \Leftrightarrow & \aleph_s(a) = \alpha & \\
\mathcal{M}, s \vDash \mathsf{DONE}_a\, \alpha & \Leftrightarrow & \aleph_u(a) = \alpha & \textit{for some } u < s \\
\mathcal{M}, s \vDash \mathsf{NEXT}\, A & \Leftrightarrow & \mathcal{M}, s{+}1 \vDash A & \\
\mathcal{M}, s \vDash \mathsf{ALWAYS}\, A & \Leftrightarrow & \textit{for all } t \geq s : \mathcal{M}, t \vDash A & \\
\mathcal{M}, s \vDash A \;\mathsf{UNTIL}^*\; B & \Leftrightarrow & \textit{if exists } t \geq s : \mathcal{M}, t \vDash B & \\
 & & \textit{then for all } s \leq u < t : \mathcal{M}, u \vDash A & \\
 & & \textit{else for all } v \geq s : \mathcal{M}, v \vDash A &
\end{array}
$$

Before we introduce the extensions to $\mathscr{L}_{TP}$ that are necessary for expressing the declarative part of the norms, we introduce a reduced logic to be used for the sub-formulas of the deontic expressions. If such a restriction is not made, expressions like 'a is obliged to see to it that b is permitted that ...', which could lead to complex nested deontic expressions which we are not considering in this work. To restrict the expressiveness of the deontic expressions, we will only allow the use of *non-temporal* operators in deontic expressions (which excludes the use of deontic operators as well, as can be seen below). To this extent we introduce a restricted logic $\mathscr{L}_{TP^-}$ which is a subset of the total logic $\mathscr{L}_{TP}$ as follows.

Definition 4.4 ($\mathscr{L}_{TP^-}$: Reduction of $\mathscr{L}_{TP}$)
The logic $\mathscr{L}_{TP^-}$ is a subset of the temporal, deontic logic $\mathscr{L}_{TP}$, reducing the expressiveness of $\mathscr{L}_{TP}$ by excluding the operators NEXT*,* ALWAYS*,* SOMETIME*,* UNTIL**, and* UNTIL*.*

In a sense, formulas of the reduced logic $\mathscr{L}_{TP^-}$ are related to the current and previous states only (i.e., the present and the past), without the possibility of expressing any changes in the future (which would require the temporal operators of $\mathscr{L}_{TP}$).

Now that we introduced the base temporal logic, we extend the logic to make it able to express the declarative part of norms. To achieve this we semantically introduce the deontic operators for expressing permissions (PERMITTED), obligations (OBLIGED) and prohibitions (FORBIDDEN) and show that these operators are, in fact, abbreviations of complex temporal formulas. We introduce a special set of propositions to denote when violations occur [Anderson, 1958; Meyer and Wieringa, 1993], meaning, a set of special propositions of the form $viol(a, P, D)$

is added to the base logic of $\mathscr{L}_{TP}$. Semantically these propositions work as any other proposition and intuitively denote that 'a violation has occurred because a did (not) see to it that P holds with respect to deadline D'.

To handle the temporal aspects of norms, such as deadlines, we use ideas from [Broersen et al., 2004; Dignum et al., 2004; Dignum, 2004] and adapted these to be used with the temporal logic as specified above. Deadlines are norms expressing that a proposition needs to hold (this proposition can, of course, also be an action that needs to be done) before a specific event (or a specific moment in time) has occurred (the deadline). This means that a violation will occur only if the action was not performed before the specified deadline. This can be captured in the following semantical definition.

Definition 4.5 (Deadlines)
For a model $\mathcal{M}$ with $a \in$ Agents, $s, t, u, v, w \in \mathbb{N}_0$ and P, D formulas of $\mathscr{L}_{TP^-}$, we define

$$\begin{aligned}
&\mathcal{M}, s \models \mathsf{OBLIGED}(a, P\ \mathsf{BEFORE}\ D) \quad \Leftrightarrow_{def}\\
&\qquad \exists t \geq s : \mathcal{M}, t \models D \textit{ and}\\
&\qquad (\forall s \leq u < t : \mathcal{M}, u \models \mathsf{NOT}\, viol(a, P, D)) \textit{ and}\\
&\qquad ((\exists s \leq v < t : \mathcal{M}, v \models P \textit{ and } \mathcal{M}, v \models \mathsf{ALWAYS\ NOT}\, viol(a, P, D)) \textit{ or}\\
&\qquad (\forall s \leq w < t : \mathcal{M}, w \not\models P \textit{ and } \mathcal{M}, t \models viol(a, P, D)))
\end{aligned}$$

This definition captures the two cases possible with deadlines; either the deadline is acted upon (i.e., the obliged proposition holds before the deadline occurs) and no violation occurs, or the proposition never holds before the deadline, thus violating the norm (i.e., a violation occurs). These two cases are expressed in definition 4.5 as follows: a) a deadline always occurs at some time, b) until that moment no violations can occur because of the norm, and c) either the obliged proposition holds before the deadline and no violation of the norm will ever occur, or the deadline passes while the proposition did not hold and a violation occurs.

An LTL-reduction of the concept of deadlines semantically defined in definition 4.5 can be given, expressing the notion of a deadline in operators already in the framework.[3]

Proposition 4.6 (LTL-Reduction of Deadlines)

$$\begin{aligned}
&\models \mathsf{OBLIGED}(a, P\ \mathsf{BEFORE}\ D) \quad \mathsf{EQUIV}\\
&\qquad \mathsf{SOMETIME}\, D\ \mathsf{AND}\ \big[(\mathsf{NOT}\, D\ \mathsf{AND}\ \mathsf{NOT}\, P\ \mathsf{AND}\ \mathsf{NOT}\, viol(a, P, D))\ \mathsf{UNTIL}\\
&\qquad ((\mathsf{NOT}\, P\ \mathsf{AND}\ P\ \mathsf{AND}\ \mathsf{ALWAYS\ NOT}\, viol(a, P, D))\ \mathsf{OR}\\
&\qquad (D\ \mathsf{AND}\, viol(a, P, D)))\big]
\end{aligned}$$

The LTL-reduction contains the same elements as the definition of a deadline: the deadline D will occur sometime in the future, and, until either the norm is fulfilled

[3]A proof of the reduction from definition 4.5 to the LTL-formula presented in proposition 4.6 can be found in Appendix A.

(proposition P has become true and the norm cannot be violated anymore) or the norm is violated (P has not been true and a violation will occur at the time D holds), it holds that deadline D has not been met, proposition P has not been true and no violation of the norm has occurred (yet).

In a similar fashion we introduce temporal prohibitions, which are prohibitions that hold up to a certain moment in time. Intuitively this would mean that until a certain moment seeing to it that a proposition holds would lead to a violation. However, it could be the case that this specific moment never occurs, which would mean that the forbidden proposition should never hold.

Definition 4.7 (Temporal Prohibitions)
For a model $\mathcal{M}$ with $a \in \mathsf{Agents}$, $s, t, u, v, w \in \mathbb{N}_0$ and P, D formulas of $\mathscr{L}_{TP^-}$, we define

$$
\begin{aligned}
&\mathcal{M}, s \vDash \mathsf{FORBIDDEN}(a, P\,\mathsf{BEFORE}\ D) \quad \Leftrightarrow_{def}\\
&\qquad \text{If } \exists t \geq s : \mathcal{M}, t \vDash D \textit{ then}\\
&\qquad\quad (((\exists s \leq u < t : \mathcal{M}, u \vDash P \textit{ and } \mathcal{M}, u \vDash \ \mathsf{NEXT}\ \mathit{viol}(a, P, D)) \textit{ and}\\
&\qquad\qquad (\forall s \leq w < u : \mathcal{M}, w \nvDash P \textit{ and } \mathcal{M}, w \vDash \ \mathsf{NOT}\ \mathit{viol}(a, P, D))) \textit{ or}\\
&\qquad\quad (\forall s \leq v < t : \mathcal{M}, v \nvDash P \textit{ and } \mathcal{M}, v \vDash \ \mathsf{NOT}\ \mathit{viol}(a, P, D) \textit{ and}\\
&\qquad\qquad \mathcal{M}, t \vDash \ \mathsf{ALWAYS\ NOT}\ \mathit{viol}(a, P, D)))\\
&\qquad \text{and if } \forall t \geq s : \mathcal{M}, t \nvDash D \textit{ then } \forall u \geq s : \ \textit{if } \mathcal{M}, u \vDash P \textit{ then}\\
&\qquad\qquad \mathcal{M}, u \vDash \ \mathsf{NEXT}\ \mathit{viol}(a, P, D)
\end{aligned}
$$

This definition captures the following intuitive meaning: a) if at sometime in the future D occurs, no violation has occurred and will occur until either P holds before D passes (after which a violation will occur) or D passes without P ever holding (and no violation of the norm occurs), or b) if D will not occur ever, a violation will occur whenever P holds. Note that we can use this definition for expressing non-temporal prohibitions (i.e., 'it is always forbidden that ...') as well; they are expressed in the second part of the definition given above. A temporal prohibition becomes a generally lasting prohibition, if the deadline specified for the temporal prohibition will never occur. One deadline guaranteed to never occur is the expression **falsum**.

Definition 4.8 (Prohibitions)
In all models $\mathcal{M}$ with $a \in \mathsf{Agents}$ and $s \in \mathbb{N}_0$ for a formula P of $\mathscr{L}_{TP^-}$ it holds that

$$\mathcal{M}, s \vDash \mathsf{FORBIDDEN}(a, P) \quad \Leftrightarrow_{def} \quad \mathcal{M}, s \vDash \mathsf{FORBIDDEN}(a, P\,\mathsf{BEFORE\ falsum})$$

LTL-reductions of temporal prohibitions (and at the same time, non-temporal prohibitions) can be given to express the prohibitions in operators already in the framework.[4]

[4] Again, a proof of the reduction from definition 4.7 to the LTL-formula presented in proposition 4.9 can be found in Appendix A.

Proposition 4.9 (LTL-reduction of Temporal Prohibitions)

$$\begin{aligned}
&\mathsf{FORBIDDEN}(a, P\ \mathsf{BEFORE}\ D) \quad \mathsf{EQUIV}\\
&\quad\Big(\mathsf{SOMETIME}\ D\ \mathsf{IMPLIES}\\
&\qquad\Big[(\mathsf{NOT}\ D\ \mathsf{AND}\ \mathsf{NOT}\ P\ \mathsf{AND}\ \mathsf{NOT}\ viol(a,P,D))\ \mathsf{UNTIL}\\
&\qquad((D\ \mathsf{AND}\ \mathsf{NOT}\ P\ \mathsf{AND}\ \mathsf{ALWAYS}\ \mathsf{NOT}\ viol(a,P,D))\ \mathsf{OR}\\
&\qquad(P\ \mathsf{AND}\ \mathsf{NEXT}\ viol(a,P,D)))\Big]\Big)\ \mathsf{AND}\\
&\quad\Big(\mathsf{ALWAYS}\ \mathsf{NOT}\ D\ \mathsf{IMPLIES}\ \mathsf{ALWAYS}\ \Big[P\ \mathsf{IMPLIES}\ \mathsf{NEXT}\ viol(a,P,D)\Big]\Big)
\end{aligned}$$

Before we introduce the semantics for permissions, let us first look at conditional norms. Conditional norms, in the classical sense, mean that whenever the condition holds, the norm is applicable. This would mean, in the case of deadlines, that whenever the condition holds before the deadline (while P has not been true) the obligation (to make P true) is applicable. In the case of temporal prohibitions it means that the condition being true before the temporal constraint has passed 'activates' the temporal prohibition.

Definition 4.10 (Conditional Deadlines and Prohibitions)
In all models $\mathcal{M}$ with $a \in$ Agents and $s \in \mathbb{N}_0$ for formulas P, D, C of $\mathscr{L}_{TP^-}$ it holds that:

$$\begin{aligned}
&\mathcal{M}, s \models \mathsf{OBLIGED}((a, P\ \mathsf{BEFORE}\ D)\ \mathsf{IF}\ C) \quad \Leftrightarrow_{def}\\
&\quad \mathcal{M}, s \models \mathsf{ALWAYS}\ \Big[\Big(C\ \mathsf{IMPLIES}\ \mathsf{OBLIGED}(a, P\ \mathsf{BEFORE}\ D)\Big)\ \mathsf{UNTIL}\ (P\ \mathsf{OR}\ D)\Big]\\
&\mathcal{M}, s \models \mathsf{FORBIDDEN}((a, P\ \mathsf{BEFORE}\ D)\ \mathsf{IF}\ C) \quad \Leftrightarrow_{def}\\
&\quad \mathcal{M}, s \models \mathsf{ALWAYS}\ \Big[\Big(C\ \ \mathsf{IMPLIES}\ \mathsf{FORBIDDEN}(a, P\ \mathsf{BEFORE}\ D)\Big)\ \mathsf{UNTIL}^*\ D\Big]
\end{aligned}$$

One can see that these definitions obtain the desired results. The first equivalence denotes, intuitively, that conditional deadlines are considered to be deadlines being triggered by the condition, but only if the condition holds before either the deadline has passed or the proposition has become true. The second equivalence expresses, intuitively, that conditional temporal prohibitions are considered to be temporal prohibitions triggered by the condition mentioned, as long as that condition holds before the temporal constraint holds, i.e. P is only prohibited as longs as the condition C of the prohibition occurs prior to the temporal constraint D.

For the interpretation of permissions, we use the classical one as presented in, e.g., [Meyer and Wieringa, 1993], meaning that being permitted that P expresses that P holding does not lead to a violation. Here, however, we introduce conditional permissions and will show that unconditional permissions can easily be obtained from those.

Definition 4.11 (Conditional Permissions)
For all models $\mathcal{M}$ with $a \in \mathsf{Agents}$ and $s \in \mathbb{N}_0$ for formulas P, C of $\mathscr{L}_{TP^-}$ it holds that:

$$\begin{aligned}\mathcal{M}, s \models \mathsf{PERMITTED}(a, P \;\mathsf{IF}\; C) \quad &\Leftrightarrow_{def} \\ &\forall t \geq s : \mathcal{M}, t \models C \Rightarrow \;\; \textit{if}\; \mathcal{M}, t \models P \;\textit{ then }\; \mathcal{M}, t \models \mathsf{NOT}\, viol(a, P)\end{aligned}$$

This definition clearly expresses that whenever the condition holds, the permitted proposition will not raise a violation. This definition can be used to express unconditional permissions as well.

Unconditional permissions differ from the conditional permissions, as defined in definition 4.11, in the fact that the hold all the time, i.e., unconditional permissions are not triggered by a certain proposition being true. This can be expressed by using `truth` as the condition of a conditional permission, since this would make the permission active for all states.

Definition 4.12 (Unconditional Permissions)
For all models $\mathcal{M}$ with $a \in \mathsf{Agents}$ and $s \in \mathbb{N}_0$ for formulas P, C of $\mathscr{L}_{TP^-}$ it holds that:

$$\mathcal{M}, s \models \mathsf{PERMITTED}(a, P) \quad \Leftrightarrow_{def} \quad \mathcal{M}, s \models \mathsf{PERMITTED}(a, P \;\mathsf{IF}\, \mathsf{truth})$$

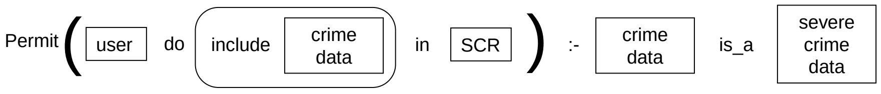
Like prohibitions and deadlines discussed before, the permissions expressed in definitions 4.11 and 4.12 can be reduced to expressions in LTL using only operators that are already in the framework.

Proposition 4.13 (LTL-Reduction of Permissions)

$$\begin{array}{ll}\mathsf{PERMITTED}(a, P \;\mathsf{IF}\; C) & \mathsf{EQUIV}\;\; \mathsf{ALWAYS}(C \;\mathsf{IMPLIES}\; (P \;\mathsf{IMPLIES}\; \mathsf{NOT}\, viol(a, P))) \\ \mathsf{PERMITTED}(a, P) & \mathsf{EQUIV}\;\; \mathsf{ALWAYS}(P \;\mathsf{IMPLIES}\; \mathsf{NOT}\, viol(a, P))\end{array}$$

The norm representation presented above covers most of the normatives used to specify normative institutions. Other normatives, such triggered obligations or prohibitions (i.e., something is obliged/prohibited *after* a certain event or action) can be expressed in the language presented, but have been omitted. Such normatives can be introduced similarly to what we have presented, but will have to be rewritten to those normatives as described by the definitions presented above for the time being.

Earlier in chapter 3.2 we used an informal 'box'-formalisation to represent the norms that crime data may be included in a severe criminality register, if it is severe criminality data as:



Using the formalism presented above, this norm can now be formally represented as:

$$\mathsf{PERMITTED}(user, \mathsf{DO}_{user}\, include(crime\_data, SCR)\ \mathsf{IF} \\ (severe\_criminality(crime\_data))$$

Roles and Role Enactment

In the previous we have introduced a declarative framework for the expression of norms, very similar to frameworks like those of [Aldewereld et al., 2005; Broersen et al., 2004; Dignum et al., 2004]. The framework allows for the expression of norms related to individual agents in the system, however, norms tend to be more general than that. Instead of specifying for every separate agent in the system what is right and wrong, norms are usually defined over groups of agents, abstracting from the individuals in these groups. These groups, denoted by *roles*, are composed of agents with similar function, position or objective in the system, [Scott, 2001]. This role of an agent, adopted upon entering the system, determines the course of actions they can undertake and which other roles they may or may not switch to, where it are these restrictions imposed on an agent by the adopted role that are exactly what is specified by the norms (as mentioned earlier in subsection 2.4.1).

In the end, though, it is the individual agents that have to abide to the norms, possibly violating them, and need to be held responsible accordingly. To connect the norms, directed to a specific role, to the individuals playing that role, we use a (meta-level) notion of role enactment and deactment (based on ideas from [Dignum, 2004] and [Dastani et al., 2004]). When an agent takes up a role (be it upon entering the institution or at pre-defined key points that allow switching of roles), the norms addressed to that role are copied and addressed to that particular agent.

For instance, the norm specifying that all buyers should pay their bought goods before leaving the auction-house institution is represented by:

$$\mathsf{OBLIGED}((buyer, \mathsf{DO}_{buyer}\, pay(price, item)\ \mathsf{BEFORE} \\ \mathsf{DONE}_{buyer}\, leave(auction\_house))\ \mathsf{IF}\ won(buyer, item, price))$$

This norm will become addressed to the agent *Shakey*, when *Shakey* enacts the role of *buyer* upon entrance of the auction-house.

$$\mathsf{OBLIGED}((Shakey, \mathsf{DO}_{Shakey}\, pay(price, item)\ \mathsf{BEFORE} \\ \mathsf{DONE}_{Shakey}\, leave(auction\_house))\ \mathsf{IF}\ won(Shakey, item, price))$$

Naturally, the general norm (addressed to the role, not the individual) remains intact and can be used when other agents enact the role of *buyer* in the institution (if multiple instantiations of the role are allowed by the institution's specification).

In the norm frame presented below, we only speak of general norms directed at roles, not at specific agents. Note, however, that all aspects of the norm frame are instantiated upon role enactment, not just the declarative representation of the norm.

4.2.2 An Annotated Approach to Operationalising Norms

The norm condition from the previous section only expresses the declarative meaning of the norm and tells us nothing about how the norm should be enforced; it can be used to derive when the norm is violated, but it does not express what should be done to prevent such a violation, or what should be done when the norm is violated (e.g., what punishments must be exerted on the violating agent). Therefore, we extend the declarative norm representation of the previous section with operational aspects to capture the operational meaning of the norm. This gives us the following norm frame (based on previous work presented in [Vázquez-Salceda et al., 2004, 2005]):

Definition 4.14 (Norms)

$$\begin{array}{rcl}
\mathsf{NORM} & := & \mathsf{NORM\_CONDITION} \\
& & \mathsf{VIOLATION\_CONDITION} \\
& & \mathsf{DETECTION\_MECHANISM} \\
& & \mathsf{SANCTION} \\
& & \mathsf{REPAIRS} \\
\mathsf{VIOLATION\_CONDITION} & := & formula \\
\mathsf{DETECTION\_MECHANISM} & := & trigger > \mathsf{PLAN} \\
\mathsf{SANCTION} & := & \mathsf{PLAN} \\
\mathsf{REPAIRS} & := & \mathsf{PLAN} \\
trigger & := & formula \\
\mathsf{PLAN} & := & action\,expression \mid action\,expression\,;\,\mathsf{PLAN}
\end{array}$$

The main ingredient of a norm is the declarative aspect, namely the *norm condition* (expressed in $\mathscr{L}_{TP}$) that we introduced in section 4.2.1. This declarative definition of the norm is annotated with the fields to express when the norm is violated (*violation condition*), how the violation is best detected (*detection mechanism*), and what the responses of the institution to norm violations should be (*sanctions* and *repairs*). Let us look at these elements in more detail.

Violation Condition

The first field added to the declarative representation of the norm is the violation condition, which is closely related to the declarative representation of the norm.

The violation condition is a formula expressed in $\mathscr{L}_{TP^-}$ that is denoting the state when the norm is violated. Every norm, as expressed in the norm condition field, implicitly defines the following stages in which a norm can be:

a) the activation of the norm,

b) the violation of the norm, and

c) the deactivation of the norm.

The activation of the norm (a) is the part of the norm that tells us when the norm is active, e.g., take the norm that every item won has to be paid for:

$$\begin{array}{l}\mathsf{OBLIGED}((buyer, \mathsf{DO}_{buyer}\, pay(price, item)\ \mathsf{BEFORE}\\ \quad \mathsf{DONE}_{buyer}\, leave(auction\_house))\ \mathsf{IF}\ won(buyer, item, price))\end{array}$$

This norm only becomes active at the moment that *buyer* wins the auction for *item* (at which time *buyer* is obliged to pay for it before leaving the auction house). This concept of activation is necessary for determining whether the norm has been violated (b), since no violations (*buyer* left without paying) can occur if the norm is not active (*buyer* did not win anything).

The deactivation of a norm (c) occurs when either the norm's activation does not hold anymore or when the norm has been fulfilled. In the case of the norm mentioned above this is the state where *buyer* pays *price* for *item*. This action deactivates the obligation, i.e., *buyer* does not have to pay *price* anymore before leaving the auction house, as he already did.

Violations of this norm (and for all obligations in general) occur only when the activation has occurred, the deactivation has not, and the obliged proposition has not been satisfied after the activation and before the deactivation. This translates to the following *violation condition* for the norm mentioned above:

$$\mathsf{NOT\ DONE}_{buyer}\, pay(price, item)\ \mathsf{AND} \tag{4.1}$$
$$\mathsf{DONE}_{buyer}\, leave(auction\_house)\ \mathsf{AND} \tag{4.2}$$
$$won(buyer, item, price) \tag{4.3}$$

In this violation condition it is stated, as one can clearly see, that the activation of the norm must have passed (4.3), the deactivation has not passed (4.2), and the obliged proposition has not been satisfied (4.1) leading to the violation of the norm.

Similarly, for prohibitions the violation condition contains an activation part, a deactivation part and a part defining the actual violation. For instance, consider the following prohibition, expressing that it is forbidden to extract a patient's organs (post-mortem, for transplantation purposes) if there is a suspicion that the patient died of non-natural causes and the coroner has not done an autopsy to verify or deny this:

$$\mathsf{FORBIDDEN}((doctor, DO_{doctor}\, extract(organ, patient)\ \mathsf{BEFORE}$$
$$\mathsf{DONE}_{coroner}\, autopsy(patient))\ \mathsf{IF}\ suspicion(non\text{-}natural\_death, patient))$$

From this prohibition we can derive the following violation condition:

$$\mathsf{DONE}_{doctor}\, extract(organ, patient)\ \mathsf{AND} \tag{4.4}$$
$$\mathsf{NOT\ DONE}_{coroner}\, autopsy(patient)\mathsf{AND} \tag{4.5}$$
$$suspicion(non\text{-}natural\_death, patient) \tag{4.6}$$

As stated, and similar to the violation condition of obligations mentioned above, we can see that the violation condition includes all three parts to determine that: a) the norm is active (4.6), b) the norm has not been deactivated (4.5), and c) a violation of the norm is occurring (4.4). Violation conditions of unconditional prohibitions, or atemporal prohibitions, are similar to the one mentioned but, respectively, loose the activation (the norm is active at all times) or the deactivation (after activation, the norm will be active for all times) part of the violation condition.

The violation condition of permissions is a bit different from the violation conditions of the prohibitions and obligations as mentioned above. Permissions, in the strict declarative use as defined above, can never be violated, and would thus have no violation condition (or have a violations condition that never holds, e.g., falsum). For example, let us regard a permission as the following, which expresses that it is permitted to take measures to preserve the (needed) organs of a deceased patient who left no statement of intent (which would grant or deny the permission for extracting the organs for transplantation) if the procedure (dictated by law) to obtain the necessary permission has not ended:

$$\mathsf{PERMITTED}((doctor, \mathsf{DO}_{doctor}\, take\_measures\_for\_preserving(organ)$$
$$\mathsf{BEFORE}\, ended(procedure\_for\_obtaining\_permission))$$
$$\mathsf{IF\ NOT}\, exists(statement\_of\_intent))$$

So the declarative representation of this permission tells us that if no statement of intent exists and the procedure for obtaining the permission for the extraction of the organs has not ended, taking measures to preserve the organs does not cause any violations. However, what if these measures are taken when a statement of intent does exist (in particular, one that denies the permission for extraction) or if the procedure has ended already (and again, in particular, when the outcome of the procedure was a denial of the permission to extract the organs)? This specific norm says nothing about these situations, and, although we would intuitively like to infer that in the cases just mentioned violations might arise, we cannot be completely sure about it (it has to be specified by other norms, such as a prohibition to extract the organs and take measures to preserve them if the permission for extraction is denied in the statement of intent of the patient).

However, if we can assume that the permission and prohibitions specified for a specific domain are complementary, we can specify a clear and concrete violation condition for permissions. The violation condition would work opposite to those for obligations and prohibitions as the activation of the permission would now serve as the deactivation of the violation condition (if the permission becomes true, no violations of the complementary prohibition can occur until the permission is deactivated again). This would lead to the following violation condition for the permission specified above:

$$\mathsf{DO}_{doctor}\, take\_measures\_for\_preserving(organ)\ \mathsf{AND} \tag{4.7}$$

$$\Big(ended(procedures\_for\_obtaining\_permission)\ \mathsf{OR} \tag{4.8}$$

$$exists(statement\_of\_intent)\Big) \tag{4.9}$$

As said, the violation condition for this permission is a bit different from the violation conditions of obligations and prohibitions as violations can only occur if the permission is *not* active. In the violation condition specified above we can see that the permission is violated[5] if: 1) the permission has not been activated (4.9) or 2) the permission has been deactivated (4.8) and 3) the assertion of the permission holds (4.7).

The specification above leads us to the following formal definition of violation conditions:

Definition 4.15 (Violation Condition)
The violation condition VC *of a norm* $\mathscr{N}$ *is defined as a formula of* $\mathscr{L}_{TP}$*-expressing the state that the norm* $\mathscr{N}$ *is violated in concrete terms.*

Although the violation condition is closely related to the norm condition, it cannot always be directly derived from the norm condition, but has to be added manually. Look again at the violation condition we specified for the permission to undertake measures for organ preservation if no statement of intent exists and the procedure to obtain the permission for extraction has not ended (4.7, 4.8 and 4.9). Although this violation condition seems intuitively correct (this is how it would be if derived from the norm condition), there seems to be something peculiar about it. Even though it correctly covers the states that indeed are a violation to the permission (as intuitively described above), it is not correct in certain states that are not a violation. Consider, for example, the case where a doctor has been given the permission for extraction through an existing statement of intent (or through a completed procedure for obtaining the permission) and is thus taking measures for preserving the organs. According to the violation condition specified in 4.7-4.9 a violation has occurred (because 4.7 and 4.9 are satisfied), but is seems strange to classify this situation as a violation since taking measurements for preserving

[5]Naturally, as specified, it is not really the permission that has been violated, as permissions cannot be violated, but more the complementary prohibition that has been violated.

the organs is a sensible action if one is permitted to extract the organs. The problem with the violation condition specified in 4.7-4.9 is that it only looks at the complementary prohibition derived from the permission, while not taking into account other permissions that also hold for the given domain (in this case; if one has the permission to extract organs one is always permitted to take measurements for preservation). To overcome this problem a more complex formalisation of the concepts involved and a more complex violation condition are needed.

Another reason why the violation condition has to be added manually and cannot always be derived directly from the norm condition has to do with the relation between actions and states. The violation condition is, as given by definition 4.15, representing a state, however, since many norms are expressed in terms of actions, this can give a problem when the action itself cannot be checked. While we expressed violation conditions above in terms of DO and DONE (see equations 4.1-4.9), this can only be done if we assume that those actions can actually be checked in the system. If, however, it concerns actions that have no direct translation in the system, or cannot be checked, defining a violation condition in terms of DOs and DONEs does not add to the operationalisation of the norm (remember, the violation condition is added to the norm condition to make the operational meaning of the norm clear, and to facilitate the implementation of the norm). For instance, expressing the violation condition of a norm that says it is forbidden to discriminate when assigning donor organs to recipients in terms of $\mathsf{DO}_{doctor}\, discriminate$ does not really help on the implementation of this norm. Mainly, there is no direct corresponding action in the system by the name *discriminate*, so it is impossible to check. Violations of this norm can, however, be detected if the violation condition clearly describes the states that are unwanted; in this case, the situations where a donor organ was assigned to a person while others on the list of possible recipients who are more eligible, are better matches for the organ, but differ in race/age/sex have been passed over. Since this translation from the action expressed in the norm to the state described by the violation condition is not always one-one (the declarative representation can abstract from a lot of interesting important details that are needed in the violation condition), the violation condition cannot be automatically derived from the norm condition. Simply translating the actions in the norm to the results of those actions, and using those to construct the violation condition is no solid solution either. Although it might work in some cases, there are problems when the result of the action (used to describe the states that violate the norms) can be achieved in other manners than by means of the obliged/prohibited action. For instance, checking whether there is light in a room because it is prohibited to turn off the lights can lead to states being incorrectly classified as violations when, for example, there was a power shortage (but no one turned off the lights).

Although violation conditions have to be added manually they are of course still required to have a strong relation to the norm condition. A common sense requirement that must be met is that the violation condition and the norm condition cannot contradict each other, e.g., for a norm $\mathsf{OBLIGED}(a, \mathsf{DO}_a\, P)$ one cannot add the violation condition $\mathsf{DONE}_a\, P$. Moreover, the violation condition and norm condition must be about the same subject. It makes no sense to add a violation condition about speeding on highways to a norm condition representing the norms of donor transplantation procedures. The relation between the violation condition and the norm condition can, however, be best seen as a sort of contextualisation as described in subsection 3.4.1, as links are made between actions and concepts in the norms and the concrete and checkable actions/concepts actually used in the violation condition. This does limit the violation condition to only defining violations of a norm in a particular context, namely that of the institution, but it does, on the other hand, allow for an easier implementation because of the use of concrete (and more importantly, checkable) aspects instead of the abstract and vague terms used in most declarative representations of norms.

Detection Mechanism

The second field added to the declarative specification of the norm is the listing of the detection mechanisms that can be used to detect the violations of the norm. Detecting violations can be very hard, depending on whether the checks needed are done in a particular order and whether the platform allows for the use of mechanisms to facilitate making checks that are otherwise too hard or time-consuming to perform. The detection mechanism field specifies the order in which the checks are best made and lists which mechanisms (provided by the platform) can be used for detecting violations of the norm.

It is easy to see that a protocol or procedure satisfies the norm when no violations occur during the execution of the protocol. The real problem in norm checking lies, however, in determining when that violation occurs. For instance, in criminal investigations, a police officer should not have more (sensitive or private) information than needed for the investigation. So an officer is doing fine as long as no violations occur (i.e., he does not have too much information). The real problem lies in determining when the officer actually has too much information.

Before determining in which order the checks have to be made when trying to detect violations, we must have some understanding of the complexity of the check. Checks that are utterly infeasible are not helpful for determining violations (in fact, those violations will, because of the complexity of the check, almost never be detected). The detection of violations is dependent on two properties of the checks that have to be done:

a) *observability*: the conditions or actions can be checked by internal agents, given the time and resources needed;

b) *computability*: the conditions or actions can be checked within a feasible (or preferably, fast) and low cost manner.

Using these two properties, we can analyse their impact on the implementation of norm enforcement.

- **Observable and computable**: verification of all the predicates and actions can be done easily, all the time.
- **Observable, but not computable**: the check is too time/resource consuming to be done all the time. This can be solved in two manners:
  - adding extra data, structures and/or mechanisms to make it easy to verify;
  - doing the verification not all the time, but delayed, periodically or at random.
- **Non-observable, but computable**: these can be internal conditions, internal actions (like reasoning) or actions that are outside the ability of the system to be observed or detected. These cannot be checked unless other conditions or actions that are observable might be found and used to (indirectly) detect a violation.
- **Non-observable and not computable**: violations of norms dependent on checks of this category cannot be detected in the system. It is recommended to enforce such norms in other manners, e.g., by regulating the access to the non-observable, non-computable predicates/actions of the system.

A particularly interesting case is the second; norms that are not computationally verifiable directly, but are so by introducing extra resources. Clear examples of these are the continuous checking of actions and deadlines. Since it would cost too many resources to make checks for every agent in the system at all times to see whether one of the agents has done a possible illegal action (an action that might trigger a violation), mechanisms can be introduced to enhance the platform in such a way that making these checks becomes more feasible. In the case of action occurrence, such mechanism can be a **black list mechanism** of actions to check and an **action alarm mechanism** that triggers an alarm when an action on the black list is attempted to start, is running, or is done. This trigger mechanism has to do no further checks, only to make sure that the enforcing agents are aware of the occurrence of the action. Similarly, checking whether a deadline has passed can overload agents, because it requires continuous checks to determine whether the deadline has passed. A **clock trigger mechanism** can be used to send a signal to the enforcer to notify them of the passage of the deadline. The idea is to implement the clock mechanism as efficiently as possible to avoid burden on the agents.

Next to the complexity of the checks needed for determining whether a violation has occurred, the order in which these checks are made can influence whether the actual enforcement will be computationally feasible. For instance, it is maybe easier to check whether an action, prohibited in some cases, has been performed before actually checking whether the norm prohibiting this action was actually active. However, when that action is performed often by the agents and is norm-compliant almost every time (the prohibition is specifying a rare exceptional case), it is easier to check whether the norm is active before starting to check if the action has been performed. This preferred order of the checks can also be made clear in the *detection mechanisms* expression.

In short, the detection mechanism field of the norm can be defined as follows.

Definition 4.16 (Detection Mechanism)
The detection mechanism *of norm $\mathscr{N}$ specifies a generic plan that should be followed to determine the status of the* violation condition VC *of* $\mathscr{N}$*. This plan specifies the order of the checks and the mechanisms present on the platform to perform these checks.*

In general, the detection mechanism lists the procedure necessary for determining whether the violation condition holds at a certain moment. Executing the plan includes performing the actions necessary to get closure on whether the violation condition holds or not, i.e., after the plan in the detection mechanism has been performed it is known that VC holds or that VC does not hold. The detection mechanism is not performed all the time, since this would be far to resource consuming. Instead, the execution of the detection mechanism is triggered through the mechanisms mentioned above, which means an action being done, a deadline ending or a predicate becoming true triggers the enforcers to check whether a violation has occurred. For instance, the detection mechanism for a norm like the following[6].

$$\begin{array}{l}\mathsf{OBLIGED}(user, \mathsf{DO}_{user}\, include(source(suspect\_data), criminal\_register)\ \mathsf{IF}\\ \quad \mathsf{DONE}_{user}\, include(suspect\_data, criminal\_register))\end{array}$$

would be triggered by an action alarm set on the $include(data, criminal\_register)$ action, after which it is checked (by the enforcers) whether the source of the included data is included in the register as well. The trigger and the checks included in the detection mechanism are derived from the norm condition and violation condition of the norm.

Sanctions and Repairs

The last two fields of the norm representation define the reactions that the system has to take when a violation has occurred. These reactions are separated into two categories, *sanctions* and *repairs*. Sanctions are actions performed by the system (or its norm enforcers) to punish the agents violating the norms, giving them an inherent deterring effect. This deterring effect of sanctions is meant to discourage agents from taking actions that are considered illegal by the institution, and try to make agents less willing to perform the actions that violate the norms, [Grossi et al., 2006a]. To achieve the discouraging effect on the agents, sanctions should

[6]This norm, extracted from the domain of Police Registers, expresses that all data concerning criminal suspects should be included in the register with the source of that data.

be designed to limit the future actions of the norm-violating agents. For instance, in an electronic auction house, fining agents that violate the norms influences the possibilities of the agents, as it makes it harder for the agent to obtain all the items it required (because the agent will have less money to spend). Similarly, changes to the reputation of the agent affects the future actions of the agent, as these might influence the outcome of future negotiations and interactions of the agent.

In [Grossi et al., 2006a] it is argued that sanctions can also serve as a compensation to those most affected by the violation of the norm. In order to provide some satisfaction or compensation to those harmed by the violation, a sanction is posed on the violating agent. This means that a violating agent becomes obliged, as a sanction, to pay an amount of money to the affected agent(s) as compensation. The difference between using sanctions solely as a deterrence or including a compensational part to the sanction signifies a difference in the role of the institution when applying the sanction. Sanctions that are solely used as a discouragement are sanctions that are applied by the institution itself, and therefore only benefit the institution itself (the fines are paid to the institution, bans are applied mainly to maintain order in the system). When sanctions are, next to the deterring effect, used as a compensation to those harmed, the institution becomes a mediator instead, interacting between the agent who committed the violation and the rest of the society.

Other differences in sanctions can be made; for example, in [Vázquez-Salceda et al., 2005] a distinction is made between direct and indirect sanctions. Direct sanctions are those that influence the agent immediately and are noticeable directly. These sanctions include fines, bans and other 'corporeal' sanctions. Indirect sanctions, on the other hand, influence the agent on a kind of meta-level, such as reputation changes or trust related sanctions. The effect of these sanctions might not be noticeable immediately but can influence the agent for a longer period of time. Combinations of both types of sanctions can be used as well.

Sanctions, as proposed in, e.g., [Vázquez-Salceda et al., 2005; Pasquier et al., 2006a,b], are usually inspired by sanctions used in human institutions. Examples of these include bans, incarceration, dismissal, fines and reputation or trust influences. However, when designing an e-institution not all human sanctions make sense, e.g., incarceration has no meaning in a virtual and digital environment where agents can clone themselves.

A big problem is that no guarantees can be given whether the sanction has the right effect on all agents possibly joining the institution. To design sanctions to work for agents, assumptions have to be made; what effect will the sanction have on them? Will they re-plan and try again, or will the sanction make them regret doing an action, and try to avoid doing it in the future? In human institutions, such assumptions about the inner process of humans can be made, and such assumptions are correct most of the time (we know how most of us think, react to certain stimuli, et cetera). Sanctions applied in human institutions are based on these assumptions to work as an effective deterrent, as humans tend to dislike spending time in prison or paying fines after one has violated a norm. Even alternative punishments, such as being put under probation, which can be seen merely as a warning, work for humans, as they apply to the moral sense of the perpetrator. For agents, however, this kind of reaction is not assured. Agents joining an e-institution are programmed by different developers, making them heterogeneous in nature. This heterogeneity means that the inner workings can be very different between agents. Since one cannot assume that all agents work in a similar manner or have the same beliefs in certain situations it can be very hard to design sanctions that are really punishments for all agents.

Another aspect that plays a role in designing meaningful sanctions is the complexity of the agents in the system. We mentioned in subsection 4.1.1 the existence of *norm-aware agents*; agents that know what norms are and have a clear conception of norms, violations and sanctions. The fact of knowing that a sanction is indeed a sanction, and not an environmental reaction to the situation at hand (or the action just performed) makes it possible to punish these agents in different (more elaborate) manners. A norm-aware agent, that tries to be norm-compliant, can be punished by the mere fact that it violated the norm; violating norms is not desired for such agents, meaning the deterring effect that sanctions are meant to have is now as much a part of the agent as it is of the sanction itself. This means that 'lesser' sanctions, such as warnings and minor fines, have a greater influence on these types of agents. In case of agents not aware of the norms, however, a sanction is nothing more than a necessary causal effect of the actions prohibited by the norms; even though the agent has no conception of 'being prohibited', it does recognise the fact that certain actions/situations will always trigger a certain response (the sanction for violating the norm). To sanction agents that are not norm-aware only two real options exist:

a) limiting the future actions of the agents; or

b) executing an action on behalf of the agent.

The first option includes, but is not limited to, sanctions such as bans and fines that are meant to restrict the agent in doing actions that are needed for it to achieve its goals (the amount of money fined was actually meant to buy goods in an auction; the ban prevented the agent from bidding before the auction closed). The second kind of sanctions are those where the institution changes some information (resource) pertaining to the agent and which usually can only be changed by the agent. This might consist in changing the reputation of the agent or in paying bills on behalf of the agent, because either the agent has granted the institution this power upon entering (by signing a contract that states that the institution has the authority to issue payments on behalf of the agent in case of violations), or because the agent had to pay a deposit when it entered the institution (the deposit is then used to pay the bills and any fines that might arise).

Sanctions are defined as follows.

Definition 4.17 (Sanctions)
The sanctions *of a norm $\mathscr{N}$ are a plan (a sequence of actions) that is performed after a violation of $\mathscr{N}$ has been detected. The sanctions are directed at the agent violating $\mathscr{N}$.*

On a higher level of analysis, one can say that sanctions are a negative reward to push agents into avoiding the states that are considered illegal. The 'reward' in this case is actually gained by not doing the illegal action, or avoiding the unwanted state, where the reward is the fact that the punishment is not being applied (i.e., if one does what is legal one will be rewarded by not being punished). In the long run, sanctions can be seen as a positive incentive to influence the agents towards the desirable states. This look on sanctions corresponds with the idea of sanctions being a deterrence, i.e., directing agents to try to remain within the acceptable states. Moreover, it also shows the relation between sanctions and the norm condition of $\mathscr{N}$, since it is exactly the norm condition that defines what those acceptable states are.

The other reaction of the system to violations of the norms is the repairs. Repairs are a sequence of actions meant to undo (or rather correct) the negative side-effects of a violation of a norm. Violations occur in a system through actions (or maybe through inaction) of the agents, putting the violating agent in an undesirable state. The recovery from this state comes from the punishment that the agent received (as specified in the sanctions), since, as mentioned above, the sanctions are in fact a positive push towards positive states (the agent is being put back on the right track). However, the action (or inaction) that leads to the violation could have undesirable side-effects that affect other agents in the system or maybe even the system as a whole. These negative consequences of the (in)action, or negative externalities of the action as they are called in [Coleman, 1990], are corrected by the sequence of actions that is represented in the repairs field.

In accordance with this specification, the definition of repairs is the following.

Definition 4.18 (Repairs)
The repairs *of a norm $\mathscr{N}$ are a plan (a sequence of actions) that describes how the negative side-effects from the occurrence of a violation of $\mathscr{N}$ can be undone.*

Let us explain the workings of the sanctions and repairs (and the added value from having the repairs) through the following example. Let us assume an institution that functions as an auction house, where an agent c bids on an item i and wins the auction. However, c does not have any money (or at least not enough) to pay for item i, which is, in itself, a violation (once an agent wins an auction, the agent is expected to pay for it). When the violation is detected (c cannot pay for i, or worse, tries to leave without paying) an unwanted situation has arisen that has to be resolved. First the responsible agent, c in this case, has to be punished. The sanction of the violated norm has to be applied by the enforcers to achieve

NORM_CONDITION	FORBIDDEN$(allocator,$ DO$_{allocator}$ $assign(organ, recipient))$ IF NOT DONE$_{hospital}$ $ensure\_compatibility(organ, recipient))$
VIOLATION_CONDITION	NOT DONE$_{hospital}$ $(ensure\_compatibility(organ, recipient))$ AND DONE$_{allocator}$ $(assign(organ, recipient))$
DETECTION_MECHANISM	$detect\_alarm(assign, 'starting') >$ $check($DONE$_{hospital}(ensure\_compatibility(organ, recipient)))$;
SANCTION	$inform(board,$ "NOT DONE$_{hospital}$ $(ensure\_compatibility(organ, recipient))$ AND DONE$_{allocator}$ $(assign(organ, recipient))$")
REPAIRS	$stop\_assignment(organ)$; $record($"NOT DONE$_{hospital}$ $(ensure\_compatibility(organ, recipient))$ AND DONE$_{allocator}$ $(assign(organ, recipient))$"$, incident\_log)$; $detect\_alarm(ensure\_compatibility, 'done')$; $check($DONE$_{hospital}$ $(ensure\_compatibility(organ, recipient)))$; $resume\_assignment(organ)$;

Figure 4.3: *Example norm.*

this result (in this case, let us say that c gets a permanent ban or some severe reputation adjustment). This is, however, not enough to solve the situation, since the item i, that has been won by c, but not paid for, cannot possibly be sold to c. This is where the repairs come into play, which are executed by the system to undo the negative side-effects of the situation: either i gets sold to the next highest bidder or is re-auctioned. After executing the repairs the state of the system has reverted to a state from which normal functioning can continue.

We already mentioned that the violation condition is loosely related to the norm condition of a norm. The repairs, on the other hand, are more closely related to the violations of the norm, as can be seen from the example and the definition of the repairs (see definition 4.18). The repairs depend on the actions that violate the norm, i.e., the actions, possibly in a certain context, that lead to the states that satisfy the violation condition. It is exactly the negative (unwanted) aspects of the actions that define what needs to be done. The repairs of a norm are, therefore, closely related to the violation condition and, indirectly, the norm condition of the norm, since these fields in the norm frame precisely define which actions will lead to violations. Using the actions derived from the violation condition, it has to be determined what the negative side-effects of these actions are to decide upon the sequence of actions represented in the repairs.

Example

Let us look at an example to clarify the norm frame of definition 4.14. The example norm shown in figure 4.3 is a norm derived from the domain of organ and tissue allocation, where, after an organ has been extracted for transplantation, the receiver of the available organ has to be determined. One step of this process is the verification of the compatibility of the organ and the intended recipient (if organ and recipient are not compatible, the organ will be rejected by the recipient's body, leading to deprecation of the organ and possibly death of the recipient). Therefore it is prohibited by law to assign an organ to a recipient before the check of the compatibility of the organ and the intended recipient is completed. This law has been represented in the (declarative) norm condition as $\mathsf{FORBIDDEN}(allocator, \mathsf{DO}_{allocator}\ assign(organ, recipient))$.

The violation condition expresses the states when the norm has been violated; in this case, the state where the assignment has been done, but the compatibility check has not. Because the norm itself is expressed in concrete terms, there is no need to refine those to define the violation condition. This violation condition is best checked by the plan described in the detection mechanism, namely, monitor the activity of the assign action (with the help of a mechanism to send an alarm when the assign action is starting), and check whether the organ that is assigned has actually been checked to be compatible with the recipient of the transplant.

The reactions of the system are specified in the sanction and repairs fields, stating that when a violation (expressed in the violation condition) has been detected (by means of the plan expressed in the detection mechanism) the board has to be informed about this unwanted/illegal situation. In this particular case, the board will review the situation and decide on an appropriate sanction, meaning that the sanctioning of the violation is actually delegated to parties outside of the (electronic) institution. The repairs, which are executed to correct the situation (to a certain extent), consist in this case of the cancellation of the illegal assignment, and waiting with continuing the transplantation process until the compatibility between the organ and the recipient has actually been verified.

4.2.3 Relation to Other Enforcement Methods

The norm frame expressed in subsection 4.2.2 defines a way of representing the norms that need to be implemented in the system. It not only expresses the meaning of the norm, but, by providing an operational meaning of the norm, also expresses how the norm should be implemented in the system. The norm frame was designed to express all aspects needed to implement norms from an institutional perspective and more specifically, by defining violations and reactions to these violations (thus allowing extended autonomy of the agents in the system, as argued in subsection 4.1.3). All elements needed for this kind of implementation are available in the norm frame:

- Defining violations: this is achieved by the *violation condition*, extended with the operational *detection mechanisms* to express how the violations are best detected

- Reactions to violations: these are expressed in the *sanction* and *repairs* fields.

Although the norm frame of subsection 4.2.2 is tailored to the violation detection and reaction methodology, it is possible to use the frame for other norm enforcement implementations as those mentioned in subsections 4.1.1 and 4.1.2. Ideally, while it is not needed for the correct working of a normative system, agents that can reason about norms can join the system, and it would be preferential that the norms of the system are understandable for those agents. This would mean that all elements of norms needed for agents to reason about norms are included in the norm frame of definition 4.14.

Agents reasoning about the norms represented in the norm frame of definition 4.14 are possible, foremost, because of the inclusion of a declarative representation of the norm (thus building on the extensive research done on deontic logics), though some of the operational annotations in the norm frame are useful for individual agents as well. For instance, the sanction field is very useful for agents reasoning about the norm as it tells them what will happen if they happen to violate the norm (thus making them able to calculate whether an unfortunate violation of the norm makes it worth the risk of using a more efficient, but not always necessarily norm-compliant, procedure for achieving their tasks). Moreover, the violation condition field can provide the agent with additional information about states that are better avoided (remember that the violation condition includes information about the activation and deactivation of the norm). Other fields, however, are not very interesting for agents reasoning about the norm. For instance, the detection mechanism field is clearly an annotation meant for the enforcement of the norm, and not needed for the agent addressed by the norm (though it does add information of interest for the enforcing agents, but that is an entirely different matter).

Although we are not trying to implement norm-aware agents, we can show that the concepts used in the norm frame can be mapped (or translated) to a more detailed architecture of norm-aware agents. For instance, the norm methodology described by López y Lopez et al. in [López y Lopez and Luck, 2002] is linked to the inner workings of their agent framework to make agents able to reason about norms and how to abide by the norms of a system. Their approach, however, also includes social norms and social commitments (norms, or rather contracts, agreed upon by individual agents, which increases their framework with elements that are not relevant to the description of institutional norms as we are describing in the previous sections). As mentioned before, the norm frame of [López y Lopez and Luck, 2002] contains the following elements: an addressee (the agent to whom the norm is directed), a beneficiary (the agent that receives benefit from compliance to the norm by the addressee), the normative goal (a specification of what is supposed to be achieved or avoided), the context (the state in which the norm is active), exceptions (the states when the norm does not apply), and punishments and rewards (reactions to norm breaking or norm compliance).

All these elements (with the sole exception of 'rewards') can be found in our norm frame:

- Addressees, beneficiaries (to a certain extent, since they are not always present in institutional norms) and normative goals are part of the norm condition.

- Context and exceptions are included in the violation condition. The violation condition in our norm frame expresses when the norm is violated (which is important from our institutional perspective) and does so by using the activation and deactivation of the norm. This activation and deactivation of the norm precisely expresses those time-spans when the norm is active, which is similar (and translatable) to the concepts used in the framework of [López y Lopez and Luck, 2002].

- Punishments are represented by the sanctions in the norm frame of definition 4.14.

The rewards concept of the framework of [López y Lopez and Luck, 2002] has no related concept in our norm frame; it is a concept for positively influencing agents that abide by the norms (i.e., norm compliance is recompensed). It is the opposite of sanctioning agents that violate the norms, but is more of use in social norms and social contracts (bonuses given to agents that comply with what was arranged), than having a true meaning in the norms of an institution.

4.3 Implementing the Norm Frame

As we mentioned earlier, implementations of normative institutions (institutions based in/on norms) are currently mostly limited to the regimentation approach (see 4.1.2, first part), where norms are implemented by constraining agents to only allow legal (or norm-compliant) actions and procedures, thus eliminating the need for norm enforcement (and norm enforcers). We already mentioned (see 4.1.2, second part, and 4.1.3) that this approach has its disadvantages, being that it can be hard to design fully regimented systems for severely restrictive systems, and, moreover, that restricting agents to legal and acceptable actions and procedures severely limits their autonomy and can even effect their efficiency (see 4.1.3 for details).

In the previous section we have introduced a norm frame for expressing the declarative and operational aspects of norms needed for implementing norms by means of norm enforcement (i.e. the detection and reaction to the violations of the norms). In this section we show how the norms expressed in the norm frame of definition 4.14 can be implemented to be used in the ISLANDER formalism.

First we give a short overview of the ISLANDER formalism in section 4.3.1. Then we extend the ISLANDER formalism with the means to express violations and introduce the methods to detect and react to these violations. Finally we show in section 4.3.3 how the norms from the norm frame presented above can be related to these new mechanisms.

4.3.1 The ISLANDER Formalism: Expressing Electronic Institutions

The ISLANDER formalism, as we explained in short in section 2.4.2, is a formal framework for specifying institutions on the basis of the possible dialogical interactions between agents in the system. This formalism has proven to be well-suited for modelling practical applications. As described in [Esteva et al., 2001], an institution is represented in the ISLANDER formalism as a combination of a *dialogical framework*, a *performative structure*, and a set of *rules of behaviour*. In the following we give a basic intuition of how institutions are expressed in the ISLANDER formalism by looking at these elements in more detail.

The dialogical framework is a collection of all those contextual elements that need to be shared by all participants of the institution. Basically, it includes those elements that are needed for all kinds of interaction, including, for example, a shared ontology and shared communicational conventions. The dialogical framework defines all the conventions required to make interaction between two or more agents possible. This includes definitions of the roles of the agents (including the relations between different roles in the system), the vocabulary used in the interactions, and the manners for expressing information that is exchanged between agents. Interactions in ISLANDER are represented as multiple dialogical activities, thus limiting agent interactions to message exchanges, which are called *illocutions*. Illocutions, seen as speech acts, are represented as formulas of the form $\iota(sender, receiver, info, time)$, where ι is the performative of the illocution (e.g., $declare$, $request$, $inform$) send by $sender$ (which includes the agent's name and its role) to the $receiver$ (similar to $sender$, including the agent's name and role) with content $info$ (expressed using the vocabulary of the institution) at time $time$. Since only dialogical interaction is allowed, actions (that are not strictly dialogical) are represented as illocutions as well, for instance, saying that one is paying represents the 'physical' act of paying. An example of an illocution is $inform(john, buyer, frank, seller, pay(painting, 100), 15\colon 30)$, expressing that $john$ (in the role of $buyer$) is paying $frank$ (in the role of $seller$) 100 at half past three for $painting$.

The interaction between agents is expressed through agent group meetings, which are called *scenes*. A scene is a role-based multiagent protocol, specifying the patterns of illocutions that can occur. A scene is defined as a directed graph where each node stands for a scene state (containing the context of the interaction, i.e., the history of illocutions that have been uttered so far), with each edge connecting two states representing the illocutions that can progress the scene. The edges of a scene graph are labelled by an illocution scheme to express exactly which illocutions can advance the scene state. Illocution schemes are illocution formulas with some unbound variables, thus denoting a certain set of possible illocutions. The scenes are collected in the performative structure, combined with the different transactions between scenes. The transactions between scenes specify how agents can move from scene to scene, based on their role and prevailing commitments. The scenes and the transactions between the scenes, contained in the performative structure, are expressed in a graphical manner called a scene dependence graph; an example graph is shown in figure 4.4.

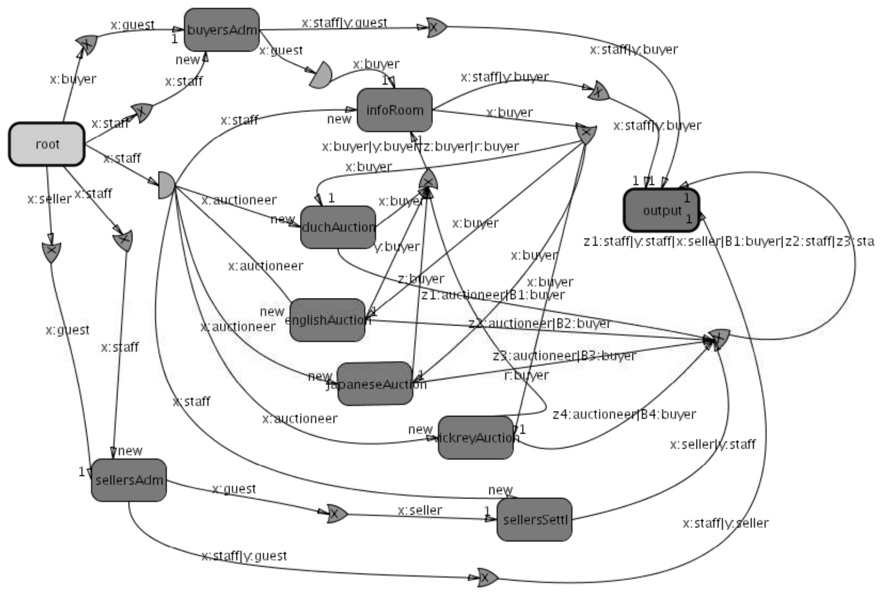


Figure 4.4: *The performative structure of an auction house institution.*

Norms in ISLANDER are either fully integrated into the performative structure or explicitly expressed in the rules of behaviour. Due to the requirement that scenes can only evolve when the correct illocutions are uttered, an implicit control of the behaviour of agents is enforced through the performative structure and scene definitions. Transitions between scene states, the results of scenes and the transitions between scenes are fully articulated in the specification of the scene and scene dependence graph, thereby restricting the possible actions of agents at any given time. For example, agents cannot pass from one scene to another if there is no explicit path of arcs in the performative structure connecting both scenes for the role or roles the agent is enacting. In a sense, the scene definition and scene dependence graph specify a protocol (though a very complicated one) which fully specifies which interaction patterns can occur in the institution and the agents are bound to following this protocol; deviations from the protocol are not possible. Next to the implicit norm implementation, ISLANDER allows for the expression of norms in the rules of behaviour, which can be seen as *norms to trigger obligations*. These norms express the consequences of some key actions (e.g., winning an auction) within the institution. The consequences are expressed as obligations that agents will acquire or satisfy depending on their illocutions within different scenes (e.g., the obligation to pay the items the agent won). These explicit norms, however, are merely constraints on the entire structure of the institution, expressing exactly what is expected of the agents, like the inter-scene transitions and the transitions between scenes. They must be fulfilled and cannot be violated; certain transitions will not be available to agents before they have completed their obligations, e.g., an agent cannot leave the auction house before it has uttered the illocution to pay the items that it had won.

As can be seen from the description above, ISLANDER only allows for the specification (and implementation, through the AMELI platform, [Esteva et al., 2004a]) of fully-regimented institutions (see section 4.1.2 for details). To achieve our goal of implementing norms by means of active norm enforcement, we first need to extend ISLANDER with the mechanisms to allow the detection of violations and the specification of reactions to such violations.

4.3.2 Institutional Mechanisms for Violation Detection/Reaction

In order to implement the norms expressed in the norm frame of definition 4.14, we first need to extend the institution with the mechanisms necessary for detecting the violations and reacting to these violations. We propose the use of *integrity constraints*, which will express when a norm is violated, and *dialogical constraints*, which will express the obligation of enforcers to react according to the violations detected, [Aldewereld et al., 2006a,b].

The integrity constraints, which we use for expressing the conditions when a norm is violated, are an idea extracted from [Esteva et al., 2004b], where they are used for the verification of norm compliancy of an institution. Integrity constraints basically express the states that should not occur in the institution. In [Esteva et al., 2004b] integrity constraints are expressed as follows:

Definition 4.19 *Integrity constraints are first-order formulas of the form*

$$\left(\bigwedge_{i=1}^{n} uttered(s_i, w_{k_i}, \boldsymbol{i}_{l_i}) \wedge \bigwedge_{j=0}^{m} e_j \right) \rightarrow \perp$$

where s_i is a scene identifier or variable, w_{k_i} is a state k_i of scene s_i (or a state variable), $\boldsymbol{i}_{l_i}$ is an illocution scheme l_i (or variable) of scene s_i and e_j is a boolean expression over variables of illocution scheme $\boldsymbol{i}_{l_i}$.

The integrity constraints express the pattern of illocutions that defines the states of the institution in which a norm has been violated. This pattern is composed of one or more utterances, which can be strengthened with additional relations that must hold over these utterances (expressed by $e_1, \ldots, e_m$). An utterance (expressed as $uttered(s_1, w_1, \boldsymbol{i}_{l_1})$) is an illocution made at a specific moment in time (i.e., in a certain scene s_1 and scene state w_1). The specific scene and scene state can be left as variables to express global utterances. The illocution in any utterance of the integrity constraint can be specified as an illocution scheme to generalise over, e.g., the agent's identity, the agent's role, details of the content of the illocution, or the time at which the illocution is uttered. Relations between utterances, for example precedence of utterances, are expressed in the additional relations. For instance, the formula:

$$\begin{aligned}&uttered(S_1, W_1, inform(a, buyer, B, R_B, content_1, T_1))\wedge\\&uttered(S_2, W_2, inform(a, buyer, C, R_C, content_2, T_2)) \wedge T_1 < T_2\end{aligned}$$

denotes the pattern where illocution the $inform(a, buyer, B, R_B, content_1, T_1)$ was uttered before the illocution $inform(a, buyer, C, R_C, content_2, T_2)$ by agent a paying role *buyer*.[7]

The integrity constraints from [Esteva et al., 2004b], expressed in definition 4.19, however, are not directly usable for the implementation of violation detection and reaction mechanisms, since these constraints express a necessary truth. Although useful for the analysis of the integrity of an institution, the integrity constraints expressed in 4.19 are too strong for the 'flagging' of violations. The constraints from definition 4.19 express which states should not occur, but also prevents these states from happening. Definition 4.19 defines a *necessary constraint*, instead of a *deontic constraint*, [Meyer et al., 1989; Meyer and Wieringa, 1993]. The difference between necessary constraints and deontic constraints is that the former express an *absolute truth*, i.e., something that cannot be violated by the system; if the left-hand side of an integrity constraint of definition 4.19 would hold, $\perp$ would have to hold, which cannot be, therefore, the left-hand side can never hold, thus restricting the possibility of the violation expressed in the constraint from ever happening. However, to denote violations that can actually occur in the system we need a weaker type of constraints, a constraint that expresses what *ideally* would hold in the system, rather then what necessarily holds in the system.

Instead, the integrity constraints, extracted from the norms, should be used by the institution to 'flag' the violations of the norms. This means that whenever an integrity constraint can be applied to the current state (i.e., the left-hand side of the constraint has been fulfilled) a 'flag' will be raised to mark that a violation of the corresponding norm has occurred. A weaker type of integrity constraint (i.e., a deontic constraint) is expressed in the following definition:

Definition 4.20 *Integrity constraints are first-order formulas of the form*

$$\left(\bigwedge_{i=1}^{n} uttered(s_i, w_{k_i}, \boldsymbol{i}_{l_i}) \wedge \bigwedge_{j=0}^{m} e_j\right) \rightarrow V_k$$

[7]In all the following we capitalise all variables used in illocution schemes, and use capitals as scene variables and scene state variables.

where s_i is a scene identifier or variable, w_{k_i} is a state k_i of scene s_i (or a state variable), $\boldsymbol{i}_{l_i}$ is an illocution scheme l_i (or variable) of scene s_i, e_j is a boolean expression over variables is illocution scheme $\boldsymbol{i}_{l_i}$ and V_k is a (unique) violation flag.

The integrity constraints of definition 4.20 work the same as those from definition 4.19, with the exception that it allows the occurrence of the violations specified. When the pattern on the left-hand side is fulfilled, the institution flags that violation for the enforcers. Naturally, only flagging the violations does not make a good enforcement strategy; reactions to the violations must be specified and executed to make an implementation of the norm enforcement method described in the second part of subsection 4.1.2.

These reactions to the violations, specified in the norm frame of definition 4.14 by *sanctions* and *repairs*, are translated to what we call *dialogical constraints.* The dialogical constraints express the behaviour of the enforcers, i.e., telling them what to do if a violation has been detected. A dialogical constraint is expressed as follows:

Definition 4.21 *Dialogical constraints are first-order formulae of the form:*

$$\left(\bigwedge_{i=1}^{n} uttered(s_i, w_{k_i}, \boldsymbol{i}^*_{l_i}) \wedge \bigwedge_{j=0}^{m} e_j \right) \Rightarrow$$

$$\left(\bigwedge_{i=1}^{n'} uttered(s'_i, w'_{k_i}, \boldsymbol{i}'^*_{l_i}) \wedge \bigwedge_{j=0}^{m'} e'_j \right)$$

where s_i, s'_i are scene identifiers or variables, w_{k_i}, w'_{k_i} are states of scenes s_i and s'_i respectively (or state variables of scenes s_i and s'_i), $\boldsymbol{i}^*_{l_i}$, $\boldsymbol{i}'^*_{l_i}$ are illocution schemes l_i (or variables) of scenes s_i and s'_i respectively, and e_j, e'_j are boolean expressions over variables from illocution schemes $\boldsymbol{i}_{l_i}$ and $\boldsymbol{i}'_{l_i}$, respectively. These boolean expressions can include functions to check the state of the institution.

The intuitive meaning of a dialogical constraint is that when the illocution pattern on the left-hand side is satisfied (i.e., the violation of the norm has occurred), the illocutions on the right-hand side must be uttered (i.e., to sanction the violator and repair the institution). In a sense, the violation of a norm by agents within the institution makes the enforcers obliged to perform the actions to punish the violating agent, making the dialogical constraint a sort of 'obligation to enforce'. Although the enforcers use the violation flags to determine if a norm has been violated, the dialogical constraint includes the pattern defining the violation (the left-hand side of the dialogical constraint is the same as the integrity constraint for a norm) to link variables in the right-hand side pattern to the corresponding illocutions that violated the norm.

The implementation of norm enforcement using these two constraints would be as follows. First, the integrity constraints are integrated in the infrastructure of the e-institution; i.e., the platform will be extended with an interpreter for checking the integrity constraints all the time, raising flags when a norm has been violated. Second, the enforcers of the institution will be programmed to periodically check whether any flags exist, and will be equipped with the dialogical constraints to ensure that they know what to do when a violation occurs. As proposed in [Aldewereld et al., 2006b], a Prolog interpreter (very similar to the one presented in [García-Camino et al., 2005]) can be implemented that would evolve the state of enactment of an institution by adding violation marks (based on violations detected through integrity constraints) and obligations to act (based on the dialogical constraints).

4.3.3 Translating Norms to Constraints

With the means to detect violations and react to the violations defined, we need to translate the norms from the norm frame of definition 4.14 to the integrity constraints and dialogical constraints as mentioned above in definitions 4.20 and 4.21.

Before we can use norms specified in the norm frame of definition 4.14 we need to translate the abstract and vague predicates and actions into corresponding concrete utterances and scenes that are specified in the definition of the institution. The translation necessary for using highly-abstracted norms in e-institutions can be considered a contextualisation (see subsection 3.4.1), as the actions (and predicates) used in the abstract norms are linked to their corresponding meaning in the domain of the e-institution. We already addressed this issue in chapter 3.4.1, and will assume here that some translation, e.g., $\mathsf{OBLIGED}((a, \mathsf{DO}_a\, A)\,\mathsf{IF}\,C)$ into $\mathsf{OBLIGED}(utter(S, W, I)\,\mathsf{IF}\,C)$[8], can be given, taking into account that scene S and state W of the institution will correspond to the applicable state meant by the norm, and that I is an illocution performed by a to implement action A. The violation conditions, sanctions and repairs of the norms can be translated in a similar fashion into utterances and boolean expressions.

The integrity constraint is then obtained from the contextualised violation condition, while the dialogical constraint is obtained from a combination of the violation condition and the sanctions and repairs of the norm. Let us clarify this by means of an example.

Consider the norm for an electronic auction house that specifies that one has to pay for the goods won in the auctions before leaving the institution. If the norm is violated, the credit card of the violating agent (which must be given before the agent is allowed to enter the institution) will be used for paying for the goods as

[8]This concrete notation of the norms in terms of utterances and propositions is very similar to the norm language presented in [García-Camino and Rodríguez-Aguillar, 2005], which was derived from the norm frame presented earlier in this chapter.

NORM_CONDITION	OBLIGED$((buyer,$ DO$_{buyer}$ $pay(price, item)$ BEFORE DO$_{buyer}$ $leave(auction_house))$IF $won(buyer, item, price)$
VIOLATION_CONDITION	NOT DONE$_{buyer}$ $pay(price, item)$ AND DO$_{buyer}$ $leave(auction\_house)$ AND $won(buyer, item, price)$
DETECTION_MECHANISM	$detect\_alarm(leave, 'starting');$ **if** $check(won(buyer, item, price)) == true$ **then** $check($DONE$_{buyer}(pay(price, item)));$
SANCTION	$debit(price + 100, creditcard, buyer)$
REPAIRS	$credit(price, creditcard, seller)$

Figure 4.5: *Example norm of an electronic auction house.*

well as a (substantial) fine for violating the norm. This norm is represented in figure 4.5.

Next we contextualise the norm frame (all concepts used are concrete, so we only have to translate them to utterances in the institution), given that we have an institution with: a registration scene (s_1), an auction scene (s_2), and a payment scene (s_3). Transitions between scenes are possible from the entrance to s_1, from s_1 to s_2 (registration is required before participating in any auction as either buyer or seller), from s_2 to s_3 (after each auction, items won have to be paid for), and from s_3 to either s_2 or the exit (after paying, the agent can decide to participate in other auctions, or leave the institution). Roles in the institution range from buyers, sellers, auctioneers (not available to external agents) to staff members (also not available to external agents).

In this context the individual parts of the norm represented in figure 4.5 needed for the implementation of the norm enforcement become the following. The violation condition would be translated to:

$$\begin{array}{l}\neg\, uttered(payment, W_1, inform(B, buyer, P, payee, pay(Item, Price), T_2)) \wedge \\ uttered(auction, w_2, inform(A, auctioneer, B, buyer, won(Item, Price), T_1)) \wedge \\ uttered(payment, W_2, inform(B, buyer, S, staff, leave(), T_3)) \wedge \\ T_1 \leq T_2 \wedge T_1 \leq T_3\end{array}$$

The illocution pattern described by the formula above exactly represents those institutional states in which the violation condition holds (i.e., someone has won something, has not paid for it, and is about to leave the institution).

The sanction, penalising the violating agent with a sum equal to the price of the item and a fine, would become the following:

$$\begin{array}{l}uttered(S_3, W_3, inform(E, staff, B, buyer, debit(Fee, creditcard, B), T_4)) \wedge \\ Fee = Price + 100\end{array}$$

And the repair, paying the seller of the item the money for which the item was auctioned, would translate to:

$$uttered(S_4, W_4, inform(E, staff, P, payee, credit(Price, creditcard, P), T_5))$$

The integrity constraint is exactly the contextualised violation condition, resulting in the following constraint that will be checked by the institution:

$$\left(\begin{array}{c} \neg\, uttered(payment, W_1, inform(B, buyer, P, payee, pay(Item, Price), T_2)) \wedge \\ uttered(auction, w_2, inform(A, auctioneer, B, buyer, won(Item, Price), T_1)) \wedge \\ uttered(payment, W_2, inform(B, buyer, S, staff, leave(), T_3)) \wedge \\ T_1 \leq T_2 \wedge T_1 \leq T_3 \end{array}\right) \rightarrow violation_1$$

This integrity constraint expresses exactly what the violation condition of the norm expresses in figure 4.5, only then in the context of this institution; a violation of the norm occurs when a buyer B has not performed the payment utterance, while having won something (in the auction scene) and wants to leave the institution.

The dialogical constraint, which tells the enforcers (which are in this case the internal agents playing the *staff* role) what to do when a violation of this norm occurs is obtained by combining the contextualised violation condition and the sanctions and repairs:

$$\left(\begin{array}{c} \neg\, uttered(payment, W_1, inform(B, buyer, P, payee, pay(Item, Price), T_1)) \wedge \\ uttered(auction, w_2, inform(A, auctioneer, B, buyer, won(Item, Price), T_2)) \wedge \\ uttered(payment, W_2, inform(B, buyer, S, staff, leave(), T_3)) \wedge \\ T_1 \leq T_2 \wedge T_2 \leq T_3 \end{array}\right) \Rightarrow$$

$$\left(\begin{array}{c} uttered(S_3, W_3, inform(E, staff, B, buyer, debit(Fee, creditcard, B), T_4)) \wedge \\ uttered(S_3, W_3, inform(E, staff, P, payee, credit(Price, creditcard, P), T_5)) \wedge \\ Fee = Price + 100 \end{array}\right)$$

The combination of the integrity constraint, which is checked by the institution and the dialogical constraint, which is used by the enforcers to determine the reaction to violations of the norm, implements the norm specified in figure 4.5 in the institution.

4.4 Concluding Thoughts

Norms play a vital role in the specification of (electronic) institutions. They can be regarded as the ‘rules of the game’, defining what is and what is not accepted in that institution. Norms specify the order in which actions must be performed, and under which conditions certain actions are acceptable or unacceptable.

In established research on this topic much attention has been given to institutions relying on the ‘hard-coding’ of norms into the agents that will act in the institution and on fully regulating these agents to prevent them from violating the norms. Recent work, however, has taken another direction, making agents aware of the norms and deliberate about them, thus making them more flexible and useable for different (normative) contexts. Another change is in the enforcement of the norms, where it has been established that the violating of the norms does not have to be a bad thing, and allowing norm violations not only increase the autonomy of the agents in the institution, but can also increase the efficiency of the systems itself.

This new approach of implementing norms through active norm enforcement of trying to detect violations of the norms and reacting accordingly (i.e., punishing the violating agents and trying to repair the situation) has brought a problem to the representation of the norms. Norms used to be represented in a declarative manner, expressing what is right and wrong, while an operational meaning of the norm is required for the implementation of active norm enforcement. To address this representational problem we added operational meaning to the norms by annotating the norms with fields that tell enforcers what to do to detect the violations of the norm, as well as what to do when the norm violation is detected.

Representing and specifying the norm in this manner makes implementing them much easier, since the method of detecting and the required reaction to a violation can almost immediately be translated to operational constraints in the (technical) implementation of the institution and its enforcers. In fact, an automated translation from the norm frame to the operational constraints has been proposed by [García-Camino and Rodríguez-Aguilar, 2006], and an implementation of the enforcement mechanism in ISLANDER (checking the integrity constraints and acting on the dialogical constraints) has been provided by a Prolog interpreter as noted earlier.

In this chapter we have shown how the transition from norms to implementable constraints can be made, and thereby giving a generic implementation method for normative institutions. Highly-regulated systems, systems with a huge number of norms governing the behaviour of the agents, can thus be implemented by introducing the constraints and enforcement necessary for guaranteeing norm compliance (with the addition of some regimenting constraints, as explained in 3.4.1 and 4.1.2). While the enforcement of the norms is guaranteed, these kinds of systems retain a very high level of freedom for agents to decide what the right course of action is to minimise the violation of the norms of the system. To assist agents in their goals, protocols can be designed that ensure that, when followed, the agent will not violate any of the norms, while still being able to (try to) achieve their (socially accepted) goals in the institution. The problem remains, however, to design such protocols, as the combined complexity of the high number of norms with the general abstraction used in the norms can make it hard to design norm compliant protocols. We will address this problem in the next chapter.

Chapter 5

Protocols for Highly-Regulated Systems

In previous chapters we introduced the means to design and implement normative institutions given their specification in terms of norms. We have shown how an active norm enforcement can be used to regulate the interactions between agents, by defining and reacting to violations. Although we consider allowing norm violations in institutions a good thing, because they increase the level of autonomy of the participating agents and can increase the efficiency of the system as a whole, the violations themselves, however, are still something that should be avoided if possible. In chapter 4 we have introduced the means to control and regulate violations that occur in the system, by applying punishments to those agents that violate the norms.

Operating in these systems that are governed by a large number of norms can be hard, particularly if one is trying to avoid violating the norms of the system. To help agents function in such a highly-regulated domain protocols can be used as guidelines to tell the agents how certain tasks can be achieved with a minimal risk (or rather, no risk at all) of violating the norms. Designing such protocols is hard due to the abstract and complex nature of norms.

In this chapter we address this problem of designing protocols for domains which are regulated by a large number of (vague and abstract) norms. We show a framework that can be used to design a (prototypical) protocol in an intuitive manner. This framework uses the notion of landmarks to denote which steps (possibly dictated by the laws of the system) *should* be taken and in which order. These landmarks will be partially extracted from the norms of the system to ensure that protocols generated by this framework have a certain level of norm compliance. The landmarks extracted from the norms will be strengthened with other landmarks to denote important steps to increase the feasibility and efficiency of the protocols, while still trying to be norm-compliant.

5.1 Protocols as Guidelines

In systems specified by norms resulting from the design process discussed in previous chapters, the agents are not constrained to act in a certain manner, thereby allowing them to do whatever they want (to a certain extent). While such freedom allows for a higher level of autonomy of the agents and for innovative new ways of achieving their goals (within the legal boundaries specified by the norms), this freedom can also work counter-effective as it can be hard to decide how things are best done in that system. Particularly, agents that are not able to reason about the norms that govern the system will have a hard time achieving their desired goals without breaking any of the norms. Moreover, if the system is governed by a lot of highly abstracted norms (which is not unusual for human institutions, and possible for the electronic counterparts of such institutions) agents that do have the capacity to understand and reason about the norms will also have quite a challenge in deciding the best course of action to achieve a particular goal.

To address this problem, guidelines can be designed to help the agents perform the most basic and frequently done tasks and to help them achieve the most common goals in the highly-regulated institutions. These guidelines, or protocols, are a step-by-step description of the actions that need to be taken to achieve a certain goal (preferably with few or no violations of the norms of the system), and describing under which conditions these steps or alternative steps have to be taken.

While existing protocols (translated from the real-world counterpart of the institution) can be used, such protocols are not always present or might not be sufficient for the electronic institution, and new protocols have to be designed. This is unfortunately hard to do due to the gap that exists between the normative specification of the system and the procedural nature of the protocols. As we have shown in earlier chapters, norms tend to be defined in some form of deontic logic, in order to express the accepted (legal) behaviour through obligations, permissions and prohibitions. It is hard to connect this kind of norms with the practice as:

a) Norms in law are formulated in an abstract way, i.e., the norms are expressed in terms of concepts that are kept vague and ambiguous on purpose.

b) Norms expressed in deontic logic are *declarative* in nature, i.e., they have no *operational* semantics (they express *what* is acceptable, but not *how* to achieve it).

c) As explained in [Wooldridge and Ciancarini, 2002], in those formalisms and agent theories based on *possible worlds*, there is usually no precise connection between the abstract accessibility relations used to characterise an agent's state and any computational model. This makes it difficult to go directly from a formal specification to an implementation in a computational system.

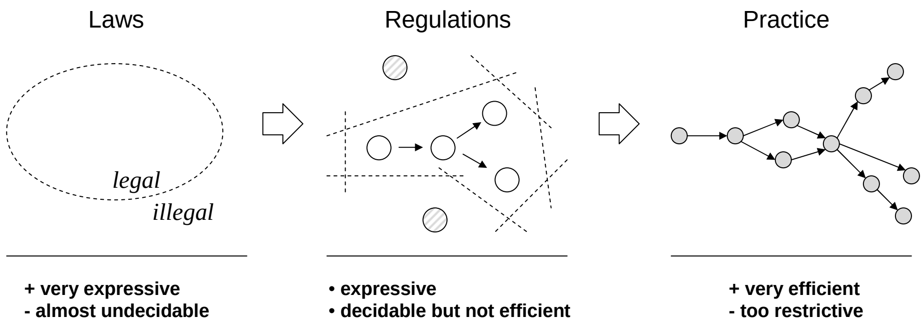


Figure 5.1: *Comparison between laws, regulations and practice.*

These three issues together create the difference between the normative and the procedural dimensions of electronic institutions (first discussed in [Dignum, 2002]), which has already been reduced from different perspectives in previous chapters. In chapter 3 we proposed formal tools to link abstract normative specifications to more concrete ones (issue a). In [Dignum et al., 2002a,b; Dignum, 2004] the expressiveness of norms (issue b) is extended by means of some variations of deontic logic that include conditional and temporal aspects [Broersen et al., 2004; Dignum et al., 2004]. However, by introducing some sort of temporal or dynamic logic operators, the resulting specification becomes more expressive but computationally too expensive to be used at run-time by agents. To solve this, the norms have to be extended with an operational meaning to express how the norms should be implemented in multiagent systems, like we have done in chapter 4. In this chapter we try to solve a part of the third issue by giving an explicit bridge between institutional norms and protocols.

5.2 From Norms to Protocols: Landmarks

Our approach is inspired by how the gap between the normative and procedural dimensions is bridged in human institutions. Human laws express in an abstract way wanted (legal) and unwanted (illegal) states of affairs. Although laws are extremely expressive, they do not express how to achieve a given state of affairs, and therefore they are exceedingly hard to use in practice to, e.g., to guide each decision in a process. In practice more efficient representations are needed, such as protocols or guidelines. In rule-based legal systems (those based in Roman-Germanic law), *regulations* add an intermediate level between laws and practice, by giving some high-level specifications on some constraints about how things can or cannot be done. These high-level descriptions are therefore interpretations of the law that add some operational constraints to be met by the practice (see figure 5.1). Using this idea, we introduce an intermediate level between institutional norm specifications and institutional protocols based on *landmarks* (the example shown in figure 5.2 will be discussed in more detail in section 5.4).

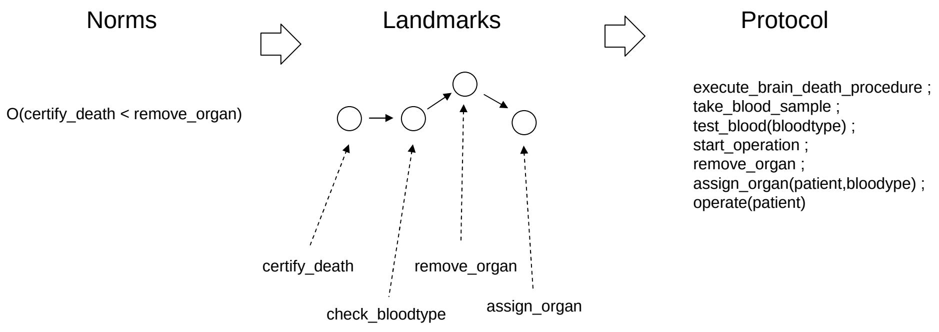


Figure 5.2: *From norms to protocols via landmarks.*

5.2.1 Landmarks

The notion of landmarks is based on work of [Smith et al., 1998], where landmarks are defined as a set of specifiable semantic properties that must hold of the agents involved (e.g., an offer has been made; an offer has been accepted). A landmark is identified by the set of propositions that are true in the state represented by the landmark. This notion of landmarks has also received much attention in recent work on multiagent systems. In [Kumar et al., 2002], for example, landmarks are used in order to specify conversation protocols between agents at an abstract level. In contrast to other approaches in the formalisation and reasoning about conversation protocols, [Kumar et al., 2002] claim that it is needed to focus on the states, or rather the effects of the conversational actions, instead of focussing on the actions themselves to do formal analysis of conversation protocols. To this end they formalise conversation protocols as a (partially ordered) pattern of states, describing the respective order in which each state should be reached (thereby abstracting from the actual means to reach these states from previous states). In [Dignum, 2004] and [Vázquez-Salceda et al., 2004] landmarks are used with similar purposes in order to provide abstract specifications of organisational interaction in general. The landmarks in [Dignum, 2004; Vázquez-Salceda et al., 2004] are formalised as state descriptions, thus being sets of states (in a modal logic setting), to model the interaction in scenes of an organisation. A landmark in this approach represents the result of an interaction script, and the (partially ordered) landmark structures that are created represent the temporal order or relation between the different stages of a scene in an organisation. The directed graph formed from the partial ordering of the landmarks is called a *landmark pattern.*

No matter how landmarks are represented – as states, or sets of states – their relevance in protocol specification is dictated by the simple observation that several different actions can bring about the same outcome. Once the outcomes of actions are organised in a structured description (i.e., a landmark pattern), it becomes possible to represent families of protocols abstracting from the actual transitions by which each protocol is constituted. This makes landmarks an ideal solution for bridging the gap between the abstract normative level and the procedural level of protocols. The landmark pattern fully captures the order in which states should occur, representing the important steps that any protocol should contain, while still abstracting from the actual procedural information on how the transition from one state to another should be achieved. In essence, a landmark pattern represents 'those steps that should be taken and in which order'.

However, this is not enough for protocol design in highly-regulated domains. In these domains explicit limitations on the procedures used by agents also exist. In fact, explicit expressions of such limitations can be found in the institutional regulations by means of norms of the prohibitive type. To capture these we, therefore, extend the notion of landmarks with *negative landmarks*, which, intuitively, mark the states that should not be reached by any protocol. By means of these negative landmarks it becomes possible to extend a landmark pattern to incorporate a reference to 'which steps should not be taken'.

The landmark pattern is then a combination of these notions of positive and negative landmarks. It expresses the states that should be achieved, the order in which these states should be achieved, as well as the states that should be avoided by all protocols designed for the domain represented in the pattern.

Definition 5.1 (Landmark Pattern)
A landmark pattern is a structure $\mathfrak{L} = \langle L^+, L^-, \leq \rangle$ *where* L^+ *and* L^- *are finite sets of landmarks and* $\leq$ *is a partial order on* L^+.

In a sense, a landmark pattern can be seen as a collection of (abstract) goals and the order in which these goals have to be achieved.[1] The negative landmarks, on the other hand, are sort of 'warning signals' to tell what is best avoided. For now, we treat landmarks as just distinct elements of a structure (the landmark pattern); a specification of what a landmark exactly is in our framework will be made clear later on.

Given by its use as an intermediate level between the norms and the protocol, a landmark pattern will have to contain *at least* the important states that are given by the normative specification of the domain. The idea is that the norm

[1]Since the protocol is designed for a specific task, there should be a landmark in the pattern that denotes the goal of the protocol (it denotes the *liveness* aspect of the protocol). This landmark is either added manually (as done in the example shown later in section 5.4, or part of the normative description (as we assume in the other examples used in this chapter). This goal of the protocol is related to the *liveness* of the protocol, while the normative landmarks are related to the *safety* of the protocol. These aspects of protocols will be discussed in more detail in chapter 6.

compliance of tasks in normative domains is bound by the order in which actions are performed by agents and the (temporal) relation between these actions and events occurring. This (temporal) relation between actions and events as specified by the norms should be captured in the landmark pattern to provide a strong (norm-compliant) basis for the protocol. For example, in the domain of organ transplantation it is essential that the permission for removal of the organs of a deceased patient is obtained before the organs are actually removed and used for organ transplantation. This collection of landmarks, extracted from the norms will be referred to as the *normative landmarks*.

Similar to the relation between norms, regulations and practice, where regulations add operational information to the restrictions given by the norms, a landmark pattern should add information to the normative goals in order to bridge the gap between the norms and the practice. The norms only give a (temporal) ordering of the states that should be reached. The normative landmark pattern extracted from these norms will thus leave many blanks, situations where the order of events/actions is undetermined by the norms or only minimally described. The choices in ordering that are not specified in the pattern, however, will in most cases influence the efficiency or even the feasibility of the pattern. For example, norms concerning organ transplantation describe that permission for organ removal must be obtained before the organs are removed[2], as well as that doctors should check whether a patient is brain death before starting the operation for removing organs of the patient. These two norms only describe that two states should be reached (*permission is obtained*, ρ, and *patient's death is certified*, δ) before another state happens (*organs are removed*, π), defining the landmark pattern $\langle\{\rho, \delta, \pi\}, \varnothing, \{(\rho, \pi), (\delta, \pi)\}\rangle$. No information is given about the ordering of states ρ and δ, but obviously making sure that δ happens before ρ is more practical then having ρ occur before δ[3]. This would mean that we add the ordering $\delta \leq \rho$ to our landmark pattern, to obtain $\langle\{\rho, \delta, \pi\}, \varnothing, \{(\rho, \pi), (\delta, \pi), (\delta, \rho)\}\rangle$.

In this case we added an extra ordering between existing landmarks to exclude the possibility of inefficient situations arising, but additional landmarks, both positive as well as negative ones, can be added to the existing pattern to express information concerning efficiency and feasibility. Take for instance an extension of our previous example, where we include the additional step of assigning the organs that were just extracted from the deceased patient. According to the law, the assignment procedure can only start after the organs have become available (though, in this case, it is in dispute whether an organ

[2]Permission for removal of organs, as described in the law on organ transplantations, is either through a statement that the deceased made before his dead, or through a complex procedure that involves the relatives of the deceased.

[3]Although mistakes in the first observation of someone being dead, before the actual medical procedure to certify that someone is brain death, seems more a thing of the past due to new procedures and technologies, mistakes can still happen and cause quite a bit of grief to relatives. The obligation to certify a patient's state of being before starting the operation to extract organs signifies that mistakes can still happen (or at least, the law rather prevents these mistakes from happening). The saying "*it is better to be safe than sorry*" comes to mind.

is available after the permission has been given to extract the organ, or after the actual extraction, for simplicity we assume the latter is the case), given in the following landmark pattern (α represents the state of *the organ being assigned*): $\langle\{\rho, \delta, \pi, \alpha\}, \varnothing, \{(\rho, \pi), (\delta, \pi), (\delta, \rho), (\pi, \alpha)\}\rangle$[4]. The procedure of assigning organs, however, needs certain information about the organ to make sure that the assigned receiver is a compatible recipient. Leaving the pattern as it currently is, this information will need to be gathered at the moment the assignment is taking place (meaning, somewhere after π, but before or at the time of occurrence of α). Most of the information needed for the assignment process (for instance, the blood type of the deceased, which influences the list of compatible recipients) can be gathered in the beginning of the protocol (after or at the moment one checks whether the deceased is medically death), and doing so can greatly improve the speed at which a organ can be assigned, because a compatible organ recipient is found earlier in the process and thus increases the chances of the success of the organ transfer (due to degration of the organ once it is recovered from the deceased patient). This means that adding a landmark (*check bloodtype*, β) to the beginning of the landmark pattern can increase the efficiency of the existing pattern. The resulting pattern would, for example, be $\langle\{\rho, \delta, \pi, \alpha, \beta\}, \varnothing, \{(\rho, \pi), (\delta, \pi), (\delta, \rho), (\pi, \alpha), (\delta, \beta), (\beta, \rho)\}\rangle$.

As can be seen from these examples, the additional landmarks to increase the efficiency of the normative landmark pattern are highly subjective, and are domain-specific (or even task-specific). As seen in the second example in particular, information from the practice, in this case the working of the procedure of the assignment process, is used to extend the normative landmark pattern to bring it closer to the operational level of protocols. Other information that can be used for this purpose includes preferred orderings of actions (e.g., rules as "normally, α is done before β", although no normative or efficiency related motivation has been given), capability related (e.g., one is only capable to include the source and reliability of recorded data if the data is actually recorded) and reachability related (e.g., reaching α when β has not happened makes it impossible to reach β).

Even though all kinds of additional information concerning efficiency and feasibility can be added to the landmark pattern, some limitations still exist that should not be overlooked. The pattern will have to satisfy certain principles to be useful at all for protocol design. Firstly, the landmark pattern needs to be norm-compliant. The addition of procedural landmarks to the normative landmark pattern should not make the landmark pattern norm violating (i.e., landmarks added, or orderings changed, should not be done when in conflict with the requirements specified by the norms[5]). Secondly, the pattern needs to contain only

[4]Note that this landmark pattern already includes the additional ordering previously discussed; the patient is checked for being brain death before the permission is obtained.

[5]Naturally this requires a norm-compliant normative landmark pattern to start with, which should in most cases be self-evident. However, domains with conflicting norms do exist, for instance due to contrary-to-duty obligations [Tomberlin, 1981; Prakken and Sergot, 1996], and

reachable landmarks, i.e., all goals specified within the landmark pattern must be achievable. The landmark pattern should not express landmarks that are unachievable by definition (e.g., a state satisfying α as well as $\neg\alpha$), or express an ordering over the landmarks that is impossible to fulfil (e.g., the ordering of a pattern where doing one part of the pattern makes it impossible to complete the other part). Lastly, next to the norm compliance and reachability restrictions, a landmark pattern should only express goals that are within the capabilities of the agents. Naturally, a landmark could represent a state that would need a cooperation of different agents to be achieved, however, states that are totally unobtainable for the agents should not occur in the pattern (e.g., the remove organs state from our earlier example, while illustrative, would be hard to reach for most (software) agents at the present day).

Although the added information about feasibility, reachability and efficiency helps a lot in closing the gap between the normative level and the protocols and makes it easier to design protocols in the end, it does make it hard to automate the entire procedure. In the following we will show that extracting the landmark pattern from a set of norms can be done semi-automatically, where the only part of this procedure that cannot be done automatically is the addition of extra landmarks to strengthen the landmark pattern extracted from the norms, mainly because these additional landmarks are highly subjective and domain-specific. We will also show how we can translate the landmark pattern into a protocol for the specified domain.

5.3 Creating Protocols by use of Landmarks

Creating a protocol for a normative domain (such as the ones described in chapter 4) is done through the use of the intermediate level of landmarks that we presented above. This process of generating a (prototypical) protocol for a normative domain is the following. First a set of landmarks is extracted from the norms governing the domain. To extract this (normative) landmark pattern from the norms we use a technique presented in [Vardi and Wolper, 1994; Vardi, 1996; Wolper, 2002], which was originally developed with model-checking in mind (some model-checkers, like SPIN, [Holzmann, 1997], are built around principles similar to those of this technique). The idea is that we create a generic, canonical-like model representing *all* LTL models that satisfy the norms of the system. This canonical model is, in fact, a finite state machine as we show later on in section 5.3.1. From this finite state machine we generate a *regular expression* expressing the characteristic features of all models satisfying the norms. We will show that this expression is, in fact, a basic landmark pattern, exactly containing all the important states (landmarks) expressed in the norms, as well as the order in which

landmark patterns containing conflicting normative goals should be solved (for instance, change the orderings, or remove normative goals, in such a way that the landmark pattern is still viable for the task at hand but norm-compliant as well).

these states must occur (thus making it a landmark pattern). This normative pattern will then be expanded with extra landmarks to strengthen it to a full landmark pattern (as described above). The landmark pattern, now including all important states, both normative-wise and efficiency-wise, will then be translated into a protocol for the normative domain[6].

Before we explain how the landmarks are extracted from the norms and are subsequently translated into a protocol, let us first look at the relations between LTL formulas and finite automata.

5.3.1 From LTL to Finite-State Automata

In order to extract the landmarks from the LTL formulas representing the norms in the domain, we need a manner of modelling the LTL formulas in such a way that their characteristic features become apparent. The choice of making a characteristic, canonical-like model for the LTL formulas by means of a finite state machine was given by the intuition surrounding LTL semantics. It is reasonable to think of a temporal formula as being a description of a set of infinite sequences; namely those that satisfy the formula. A natural way to check whether a specific sequence satisfies a temporal logic formula φ, is to attempt to label each of the states of the sequence with the subformulas of φ. One would start with labelling the states with the propositional subformulas, and then going outwards adding exactly those subformulas that are compatible with the LTL semantics. This procedure can, of course, not be applied for infinite sequences, but the intuition it presents will prove useful for the link between LTL formulas and finite automata.

The idea would then be, given the fact that LTL formulas represent infinite traces that satisfy that formula, to generate a finite state machine that accepts exactly those infinite traces. Finite-state machines that accept infinite traces are known as Büchi automata, see [Büchi, 1962; Thomas, 1990].

The temporal logic that we are going to use in the following is almost identical to the one presented in chapter 4, with the most important difference that we only allow negations to be applied to atomic propositions (all formulas of the logic are in negated normal form). The formulas of linear-time temporal logic build from a set of atomic propositions P are the following:

- **true**, **false**, p, $\neg p$ for all $p \in P$;
- $\varphi_1 \wedge \varphi_2$ and $\varphi_1 \vee \varphi_2$, where φ_1 and φ_2 are LTL formulas;
- $\bigcirc\varphi_1, \varphi_1$ **until** φ_2, and φ_1 **releases** φ_2, where φ_1 and φ_2 are LTL formulas.

[6]Note that the technique described here is not designed to generate all possible protocols, but to provide help in the creation of protocols for a normative domain, in general. Generating all protocols from a landmark pattern would be hard due to the subjective choices that can be made. The idea of our approach is more to give a 'skeleton' or 'prototypical' protocol by means of the landmark patterns, since, at least, the normative landmarks extracted from the norms must, in some way, be satisfied in every protocol for that task in a given domain.

The operator $\bigcirc$ is similar to NEXT used in chapter 4, i.e., it relates to the *next* state. Likewise, the **until** operator is similar to the UNTIL operator used in chapter 4 in that it requires the first argument to be true *until* the second argument is true, which is required to happen. The **releases** operator is the dual of **until** and requires its second argument always to be true, a requirement that is *released* as soon as its first argument becomes true. The extension of the logic of chapter 4 with a 'release' operator is necessary for the compliance to the negated normal form, as the abbreviation used in chapter 4 to define the SOMETIME-operator is not compliant with this normal form, and has to be redefined using the newly introduced **release** operator. The new definition of the *always* and *sometime* operators would now become (related to the SOMETIME and ALWAYS operators used in chapter 4):

- $\Diamond\varphi =$ **true until** φ, which is read "eventually" and requires that its argument be true at some point in the future (like SOMETIME φ in chapter 4);
- $\Box\varphi =$ **false releases** φ, which is read "always" and requires that its argument be true at all future points (like ALWAYS φ in chapter 4).

We define the semantics of LTL (like in chapter 4) with respect to sequences $\sigma : \mathbb{N} \to 2^P$.[7] For a sequence σ, we write $\sigma(i)$ to denote the i^{th} state of σ, and σ^j to denote the suffix of σ obtained through removing the first j states of σ, i.e., $\sigma^j(i) = \sigma(j+i)$. The truth value of a formula on a sequence σ, which is given as the truth value obtained by evaluating the formula on the first state of the sequence, is given by the following rules:

- For all σ, we have $\sigma \vDash$ **true** and $\sigma \nvDash$ **false**;
- $\sigma \vDash p$ for $p \in P$ iff $p \in \sigma(0)$;
- $\sigma \vDash \neg p$ for $p \in P$ iff $p \notin \sigma(0)$;
- $\sigma \vDash \varphi_1 \wedge \varphi_2$ iff $\sigma \vDash \varphi_1$ and $\sigma \vDash \varphi_2$;
- $\sigma \vDash \varphi_1 \vee \varphi_2$ iff $\sigma \vDash \varphi_1$ or $\sigma \vDash \varphi_2$;
- $\sigma \vDash \bigcirc\varphi_1$ iff $\sigma^1 \vDash \varphi_1$;
- $\sigma \vDash \varphi_1$ **until** φ_2 iff there exists $i \geq 0$ such that $\sigma^i \vDash \varphi_2$ and for all $0 \leq j < i$, we have $\sigma^j \vDash \varphi_1$;
- $\sigma \vDash \varphi_1$ **releases** φ_2 iff for all $i \geq 0$ such that $\sigma^i \nvDash \varphi_2$, there exists $0 \leq j < i$ such that $\sigma^j \vDash \varphi_1$.

[7]With respect to the semantics presented in chapter 4, definitions 4.2 and 4.3, this relates to the sequence $\mathbf{W}$, where $\mathbb{N}$ indicates the state η_i of $\mathbf{W}$, and the element of 2^P relates to the set of propositions valid in η_i as defined by v_i in $\mathbf{W}$. See section 4.2.1 for more details.

Given these semantics it can easily be shown that this is equivalent to a version of the temporal language $\mathscr{L}_{TP}$ presented in 4.2.1 where all DO and DONE operators have been flattened to pure propositional expressions (e.g., $\mathsf{DO}\,\rho$ would become the proposition $do(\rho)$, thus eliminating the operators of $\mathscr{L}_{TP}$ to denote actions).

Although many normative operators can be defined in this temporal logic (as we have shown earlier in 4.2.1, and was done, for instance, in [Broersen et al., 2004; Dignum et al., 2004; Dignum, 2004]), we limit the operators that are considered here to deadlines and (non-temporal) prohibitions. The LTL representation of these, derived from propositions 4.6 and 4.9, are the following'

$$\begin{aligned} O(\rho < \delta) \quad &\Leftrightarrow \quad \Diamond\delta \wedge \Big[(\neg\delta \wedge \neg\rho \wedge \neg v(\rho,\delta)) \ \mathbf{until} \\ &\qquad ((\rho \wedge \neg\delta \wedge \Box\neg v(\rho,\delta)) \vee (\neg\rho \wedge \delta \wedge v(\rho,\delta)))\Big] \qquad (5.1) \\ F(\alpha) \quad &\Leftrightarrow \quad \Box(\neg\alpha \vee \bigcirc v(\alpha)) \qquad (5.2) \end{aligned}$$

The obligation expressed in (5.1) is similar to the deadline expressed earlier in proposition 4.6. This deadline is a strict deadline, where ρ must occur on or more states before the occurrence of δ. The prohibition of (5.2) is directly derivable from proposition 4.9 in that it expresses a prohibition that is not temporally bound (i.e., the deadline, expressed in proposition 4.9 never occurs, making the prohibition last forever). It exactly expresses the intuitive notion of a prohibition; whenever the forbidden state α occurs, a violation occurred. Later on in this section we show how norms expressed by these LTL formulas can be converted to a finite automaton.

Automata on Infinite Sequences

The theory of automata, and thereby that of abstract computing devices, goes back as far as the 1930s to the studies of *Turing machines* by Alan Turing. Although much more complex than today's finite state machine, the Turing machine, in its objective to study and express the capabilities of computing devices, laid the foundation of the finite automata theory. The finite automata that we consider here, date back to the 1940s and 1950s, where they were studied and used to model brain functions, but turned out to be useful for a variety of purposes, one being the study of formal 'grammars', as done in the late 1950s by the linguist Noam Chomsky. A good introduction to the theory of finite automata can be found in [Hopcroft et al., 2006] and [Sipser, 1997].

The automata that we consider are Büchi and generalised Büchi automata on infinite words. Infinite words are sequences of symbols isomorphic to the natural numbers, or more precisely, an infinite word w over an alphabet Σ is a mapping $w : \mathbb{N} \rightarrow \Sigma$ ([Thomas, 1990; Staiger, 1997]). Büchi infinite word automata have exactly the same structure as traditional finite word automata, with the exception that their semantics are defined over infinite words. A Büchi automaton is defined as a tuple $A = (\Sigma, S, \Delta, S_0, F)$ where

- Σ is an alphabet,
- S is a set of states,
- $\Delta : S \times \Sigma \rightarrow S$ (deterministic) or $\Delta : S \times \Sigma \rightarrow 2^S$ (nondeterministic) is a transition function,
- $S_0 \subseteq S$ is a set of initial states (a singleton for deterministic automata), and
- $F \subseteq S$ is a set of accepting states

A word w is accepted (or recognised) by a Büchi automaton $A = (\Sigma, S, \Delta, S_0, F)$ if there exists a sequence $\lambda : \mathbb{N} \rightarrow S$ of states such that

- $\lambda(0) \in S_0$ (the initial state of λ is an initial state of A),
- $\forall i \geq 0,\ \lambda(i+1) \in \Delta(\lambda(i), w(i))$ (the sequence of states is compatible with the transition relation of A),
- $\inf(\lambda) \cap F \neq \varnothing$ where $\inf(\lambda)$ is the set of states that appear infinitely often in λ (the set of repeating states of λ intersects the accepting set F).

Generalised Büchi automata differ from Büchi automata by their acceptance condition. The acceptance condition of a generalised Büchi automaton is a set of sets of states $\mathcal{F} \subseteq 2^S$, and the requirement that some state of each of the sets $F_i \in \mathcal{F}$ appears infinitely often. More formally, a generalised Büchi automaton $A = \{\Sigma, S, \Delta, S_0, \mathcal{F}\}$ accepts a word w if there is a sequence λ of states for w by states of A that satisfies the same first two conditions as given above, the third being replaced by:

- For each $F_i \in \mathcal{F}$, $\inf(\lambda) \cap F_i \neq \varnothing$.

We use nondeterministic, generalised Büchi automata for the first stage of the translation (the reason for this will be apparent later on), which we can then translate to an equivalent nondeterministic Büchi automaton to extract the regular expression describing this automata. This translation from generalised Büchi to 'normal' Büchi will be presented later on when we discuss the relation between the automata and regular expressions.

Linking LTL to Büchi Automata

The translation from LTL formulas to (generalised) Büchi automata is taken from work presented in [Vardi and Wolper, 1994; Vardi, 1996; Wolper, 2002]. The idea is that the sequences (words) accepted by an automaton correspond exactly to those LTL sequences satisfying the formula. The problem thus becomes the following: given an LTL formula φ build from a set of atomic propositions P, construct an automaton A on infinite words over the alphabet 2^P that accepts exactly the infinite sequences satisfying φ.

Before the procedure given in [Wolper, 2002] is presented, let us first look at the problem of determining whether a sequence $\sigma : \mathbb{N} \rightarrow 2^P$ satisfies a formula φ. This is done, as presented in [Vardi and Wolper, 1994; Wolper, 2002], by labelling the sequence with subformulas of φ in a way that respects LTL semantics. In a sense, this labelling τ of sequence σ indicates which (temporal) subformulas hold at each state of σ, i.e., a subformula φ_1 of φ (it is defined below what a subformula is) labels a position i (written as $\varphi_1 \in \tau(i)$), if and only if $\sigma^i \models \varphi_1$ (that is, the sequence σ^i satisfies φ_1 in accordance with the rules given on page 110)[8]. Before this sequence labelling is defined, let us first define what a subformula of φ is. The set of subformulas of a formula φ, called the *closure* of φ ($cl(\varphi)$), is defined as follows:

- $\varphi \in cl(\varphi)$;
- $\varphi_1 \wedge \varphi_2 \in cl(\varphi) \Rightarrow \varphi_1, \varphi_2 \in cl(\varphi)$;
- $\varphi_1 \vee \varphi_2 \in cl(\varphi) \Rightarrow \varphi_1, \varphi_2 \in cl(\varphi)$;
- $\bigcirc\varphi_1 \in cl(\varphi) \Rightarrow \varphi_1 \in cl(\varphi)$;
- φ_1 **until** $\varphi_2 \in cl(\varphi) \Rightarrow \varphi_1, \varphi_2 \in cl(\varphi)$;
- φ_1 **releases** $\varphi_2 \in cl(\varphi) \Rightarrow \varphi_1, \varphi_2 \in cl(\varphi)$.

The labelling τ of a sequence σ (called a *closure labelling* in [Wolper, 2002]) denotes formulas of the closure of φ that hold at a given position.[9] To guarantee the correspondence between positions in τ with positions in σ, the closure labelling is defined by means of a set of rules that mirrors LTL semantics. This definition (taken from [Wolper, 2002]) is as follows: a closure labelling $\tau : \mathbb{N} \rightarrow 2^{cl(\varphi)}$ of a sequence $\sigma : \mathbb{N} \rightarrow 2^P$ for a formula φ over a set of atomic propositions P is valid when it satisfies the following rules for every $i \geq 0$:

1. **falsum** $\notin \tau(i)$;
2. for $p \in P$, if $p \in \tau(i)$ then $p \in \sigma(i)$, and if $\neg p \in \tau(i)$ then $p \notin \sigma(i)$;
3. if $\varphi_1 \wedge \varphi_2 \in \tau(i)$ then $\varphi_1 \in \tau(i)$ and $\varphi_2 \in \tau(i)$;

[8]Validly labelled structures such as these are called Hintikka Models in modal logic literature. In [Hintikka, 1962] Hintikka introduced a method for creating semantical models for the study of epistemic logic. The idea used there is similar to the one presented in [Wolper, 2002] which is shown here; the model gives a (partial) description of possible states of affairs, build iteratively from formulas that are already in the model.

[9]A state s in the sequence σ is an LTL state characterised in the propositional elements that hold in that LTL state. These LTL state descriptions will ultimately form the symbols of the language accepted by the automaton, the sequence of these state descriptions being the words accepted by the automaton. The closure labelling, on the other hand, denotes the temporal formulas that must at least hold at the different stages of an automaton run, they will ultimately be used as the labellings of the automaton states. The relation between the transition labels (symbols of the language) and the automaton state labels is then given by rules 1 - 8, as specified in proposition 5.4.

4. if $\varphi_1 \vee \varphi_2 \in \tau(i)$ then $\varphi_1 \in \tau(i)$ or $\varphi_2 \in \tau(i)$;

These rules ensure that the propositional part of LTL is satisfied by the closure labelling. Note that because the rules are specified as 'if' rules and not as 'if and only if' rules, the closure labelling is not required to be *maximal*, i.e., the rules give the requirements that *must* be satisfied by the closure labelling, but do not require that all formulas of the closure that hold at a given position are included in the label of that position.

For the temporal operators, the following rules are given.

5. if $\bigcirc\varphi_1 \in \tau(i)$ then $\varphi_1 \in \tau(i+1)$;
6. if φ_1 **until** $\varphi_2 \in \tau(i)$ then either $\varphi_2 \in \tau(i)$, or $\varphi_1 \in \tau(i)$ and φ_1 **until** $\varphi_2 \in \tau(i+1)$;
7. if φ_1 **releases** $\varphi_2 \in \tau(i)$ then $\varphi_2 \in \tau(i)$, and either $\varphi_1 \in \tau(i)$ or φ_1 **releases** $\varphi_2 \in \tau(i)$.

Rule 5 for the $\bigcirc$ operator follows directly from the LTL semantics of the operator. For the **until** and **releases** operators, however, the equivalences φ_1 **until** $\varphi_2 \equiv (\varphi_2 \vee (\varphi_1 \wedge \bigcirc(\varphi_1$ **until** $\varphi_2)))$ and φ_1 **releases** $\varphi_2 \equiv (\varphi_2 \wedge (\varphi_1 \vee \bigcirc(\varphi_1$ **releases** $\varphi_2)))$ are used for the definition of rules 6 and 7 instead, since they avoid the reference to a possibly infinite set of points in the sequence (it can be easily shown that these equivalences follow from the semantics of these operators).

Unfortunately, an extra rule is required to guarantee that the labelling satisfies subformulas with an **until**. Since rule 6 does not force the existence of a point at which φ_2 appears, this can be postponed forever (which is inconsistent to the LTL semantics of the **until** operator). An extra rule to guarantee the existence of this point satisfying the *eventuality* (formulas of the form φ_1 **until** φ_2 are called eventualities, because φ_2 must eventually hold) has to be added:

8. if φ_1 **until** $\varphi_2 \in \tau(i)$ then there is a $j \geq i$ such that $\varphi_2 \in \tau(j)$.

Given these restrictions, a formalisation of the relation between the closure labelling and the LTL sequence is given in [Wolper, 2002]:[10]

Theorem 5.2 *Consider a formula φ defined over a set of propositions P and a sequence $\sigma : \mathbb{N} \to 2^P$. One then has that $\sigma \models \varphi$ iff there is a closure labelling $\tau : \mathbb{N} \to 2^{cl(\varphi)}$ of σ satisfying rules 1 - 8 and such that $\varphi \in \tau(0)$.*

Given this theorem, the relation between an automaton A accepting all sequences satisfying φ and the LTL models is rather obvious. Recall that automata accept infinite sequences (words) when this sequence can be labelled by states of the automaton, while satisfying the conditions that the first state of the sequence is a start state of A, the transition relation of A is respected and the acceptance

[10]The proof of this theorem can be found in [Wolper, 2002].

condition of A is met. It then becomes obvious to use $2^{cl(\varphi)}$ as state set (and possible state labels), and create an automaton over the alphabet 2^P that satisfies the necessary properties expressed in the labelling rules expressed above.[11]

Definition 5.3 *A Büchi automaton for a formula φ is a tuple $A_\varphi = (\Sigma, S, \Delta, S_0, \mathcal{F})$, where*

- $\Sigma = 2^P$
- *S is the set of states $\mathbf{s} \subseteq 2^{cl(\varphi)}$ that satisfy*
  - $\mathbf{falsum} \notin \mathbf{s}$*;*
  - *if $\varphi_1 \wedge \varphi_2 \in \mathbf{s}$ then $\varphi_1 \in \mathbf{s}$ and $\varphi_2 \in \mathbf{s}$;*
  - *if $\varphi_1 \vee \varphi_2 \in \mathbf{s}$ then $\varphi_1 \in \mathbf{s}$ or $\varphi_2 \in \mathbf{s}$.*
- *The transition function Δ is defined as $\mathbf{t} \in \Delta(\mathbf{s}, \mathbf{a})$ iff*
  - *For all $p \in P$, if $p \in \mathbf{s}$ then $p \in \mathbf{a}$.*
  - *For all $p \in P$, if $\neg p \in \mathbf{s}$ then $p \notin \mathbf{a}$.*
  - *If $\bigcirc\varphi_1 \in \mathbf{s}$ then $\varphi_1 \in \mathbf{t}$.*
  - *If φ_1 **until** $\varphi_2 \in \mathbf{s}$ then either $\varphi_2 \in \mathbf{s}$, or $\varphi_1 \in \mathbf{s}$ and φ_1 **until** $\varphi_2 \in \mathbf{t}$.*
  - *If φ_1 **unless** $\varphi_2 \in \mathbf{s}$ then $\varphi_2 \in \mathbf{s}$ and either $\varphi_1 \in \mathbf{s}$, or φ_1 **unless** $\varphi_2 \in \mathbf{t}$.*
- $S_0 = \{\mathbf{s} \in S \mid \varphi \in \mathbf{s}\}$.

Proposition 5.4 *The Büchi automaton $A_\varphi = (\Sigma, S, \Delta, S_0, \mathcal{F})$ accepts all and only the sequences $\sigma : \mathbb{N} \to 2^P$ satisfying a formula φ*

The restriction on the states S of A_φ ensures that the states (and thus the closure labels) satisfy rule 1 as well as rules 3 and 4 specified above. Rule 2 and rules 5 - 7 are enforced in the transition function Δ of A_φ. This ensures that the automaton A_φ complies with the rules 1 - 7 of the closure labelling specified above. The restriction on the start states S_0 of A_φ ensures that φ appears in the label of the first position of the sequence (thus limiting the labellings of A_φ to those where $\varphi \in \tau(0)$, as required by theorem 5.2). To ensure rule 8, the acceptance condition of the automaton is used. Rule 8 specifies that every state that contains an eventuality $e(\varphi')$ (where φ_1 **until** $\varphi' \in cl(\varphi) \Rightarrow e(\varphi') \in cl(\varphi)$) is followed at some point by a state that contains φ'. This means that labellings where $e(\varphi')$ appears indefinitely without φ' ever appearing must be avoided. These labellings can be

[11]The alphabet of an automaton A for a formula φ consists of all possible LTL-worlds, being that an LTL-world is described as the collection of the propositions that hold in that world. For instance, if we only consider the propositions ρ, δ and γ, the LTL-world that satisfies only ρ and δ will be denoted by $\{\rho, \delta\}$ (we also use $\rho\delta$ or $\delta\rho$ to denote this world). Conversely, the label γ denotes the LTL world in which γ holds, but ρ and δ are false.

avoided by requiring that the automaton goes infinitely often through a state in which both $e(\varphi')$ and φ' appear or in which $e(\varphi')$ does not hold. To achieve this, the acceptance condition of A_φ is specified as the following generalised Büchi condition:

- If the eventualities appearing in $cl(\varphi)$ are $e_1(\varphi_1), \ldots, e_m(\varphi_m)$, then $\mathcal{F} = \{F_1, \ldots, F_m\}$ where $F_i = \{\mathbf{s} \in S \mid e_i, \varphi_i \in \mathbf{s} \vee e_i \notin \mathbf{s}\}$.

In accordance to this definition of an automaton accepting the sequences that satisfy φ, [Wolper, 2002] states a procedure to generate a minimal Büchi automaton[12] satisfying the constraints mentioned.

Generating a Büchi Automaton from an LTL Formula

The procedure mentioned in [Wolper, 2002] to generate a Büchi automaton based on an LTL formula uses the fact that all rules given for the transition relation of the automaton require that, if some subformula of φ appears in the current state, certain other subformulas of φ must appear in the current state as well or in the next state. To achieve this, [Wolper, 2002] defines an operation *saturate* that expands states with all formulas that *must* be true in the current and next state.

Definition 5.5 (Saturate) *Let* $\mathbf{Q} = \{\mathbf{q_1}, \mathbf{q_2}, \ldots, \mathbf{q_k}\} \subseteq 2^{cl(\varphi)}$*, then saturate*$(\mathbf{Q})$ *is defined as follows.*

1. *Repeat until stabilisation: for each* $\mathbf{q_i} \in \mathbf{Q}$*,*
   (a) *if* $\varphi_1 \wedge \varphi_2 \in \mathbf{q_i}$*, then*
   $\mathbf{Q} := \mathbf{Q} \backslash \{\mathbf{q_i}\} \cup \{\mathbf{q_i} \cup \{\varphi_1, \varphi_2\}\}$;
   (b) *if* $\varphi_1 \vee \varphi_2 \in \mathbf{q_i}$*, then*
   $\mathbf{Q} := \mathbf{Q} \backslash \{\mathbf{q_i}\} \cup \{\mathbf{q_i} \cup \{\varphi_1\}\} \cup \{\mathbf{q_i} \cup \{\varphi_2\}\}$;
   (c) *if* φ_1 **until** $\varphi_2 \in \mathbf{q_i}$*, then*
   $\mathbf{Q} := \mathbf{Q} \backslash \{\mathbf{q_i}\} \cup \{\mathbf{q_i} \cup \{\varphi_2\}\} \cup \{\mathbf{q_i} \cup \{\varphi_1, \bigcirc(\varphi_1 \textbf{ until } \varphi_2)\}\}$;
   (d) *if* φ_1 **releases** $\varphi_2 \in \mathbf{q_i}$*, then*
   $\mathbf{Q} := \mathbf{Q} \backslash \{\mathbf{q_i}\} \cup \{\mathbf{q_i} \cup \{\varphi_1, \varphi_2\}\} \cup \{\mathbf{q_i} \cup \{\varphi_2, \bigcirc(\varphi_1 \textbf{ releases } \varphi_2)\}\}$
2. *Remove all* $\mathbf{q_i} \in \mathbf{Q}$ *for which* **false** $\in \mathbf{q_i}$

If this operation is applied to a single element $\mathbf{q}$ of $\mathbf{Q}$, the result is a set of sets of formulas that represent the possible ways of satisfying the requirements given by the formulas in $\mathbf{q}$. The resulting set of formulas, or actually the set

[12]The automata resulting from the procedure mentioned in [Wolper, 2002], and discussed below, are built on basis of the transitions needed from the start states of the automata. Only the states that are required as targets of the transitions generated by the procedure are added, thus creating the automaton 'by need', keeping it small.

of states (remember, a state label is defined as a set of formulas), expresses the states that need to follow $\mathbf{q}$ to satisfy the temporal restrictions expressed by the formulas of $\mathbf{q}$. The most interesting elements of these sets of formulas are the propositional formulas and the formulas beginning with a $\bigcirc$ operator, since these give the restrictions that must hold in the current state (and thus the labelling of the transitions leaving that state, because these labellings are dependent on propositions that hold in the state, see definition 5.3) and the next state in the sequence. The following filters on a set of LTL formulas $\mathbf{q}$ are thus defined:

1. $X(\mathbf{q}) = \{\varphi_i \mid \bigcirc \varphi_i \in \mathbf{q}\}$ (the 'next' formulas in $\mathbf{q}$ with their $\bigcirc$ operator stripped),
2. $P(\mathbf{q}) = \{p_i \mid p_i \in \mathbf{q} \wedge p_i \in P\}$ (the atomic propositions in $\mathbf{q}$),
3. $nP(\mathbf{q}) = \{p_i \mid \neg p_i \in \mathbf{q} \wedge p_i \in P\}$ (the negated atomic propositions in $\mathbf{q}$).

We now have all the elements needed to present the algorithm for creating automata from LTL formulas, as given by [Wolper, 2002]. The alphabet and acceptance state of the automaton are taken as defined in definition 5.3, while the set of states, the starting states and the transition relation of the automaton are given by the algorithm presented below, these elements of the automaton will be generated progressively. For ease of notation, $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$, with $\mathbf{s}, \mathbf{s}' \in S$ and $\mathbf{a} \in \Sigma$, are used to represent the elements of the transition relation Δ.

Definition 5.6 (Algorithm for generating Büchi Automata)

```
build-auto($\varphi$)
1. $S := \varnothing$ ; $\Delta := \varnothing$ ; $S_0 := \{\mathbf{q} \cap cl(\varphi) \mid \mathbf{q} \in saturate(\{\{\varphi\}\})\}$
2. $unp := \{(\mathbf{q} \cap cl(\varphi), X(\mathbf{q})) \mid \mathbf{q} \in saturate(\{\{\varphi\}\})\}$
3. while $unp \neq \varnothing$ do
     Choose and remove $(\mathbf{s}, \mathbf{x})$ from unp;
     $S := S \cup \{\mathbf{s}\}$;
     For each $\mathbf{q} \in saturate(\{\mathbf{x}\})$ do
       For each $\mathbf{a} \in \Sigma$ such that $P(\mathbf{s}) \subseteq \mathbf{a} \wedge nP(\mathbf{s}) \cap \mathbf{a} = \varnothing$
         $\Delta := \Delta \cup \{(\mathbf{s}, \mathbf{a}, \mathbf{q} \cap cl(\varphi))\}$
       if $(\mathbf{q} \cap cl(\varphi), X(\mathbf{q})) \notin unp \wedge \mathbf{q} \cap cl(\varphi) \notin S$
         then $unp := unp \cup \{(\mathbf{q}, X(\mathbf{q}))\}$
```

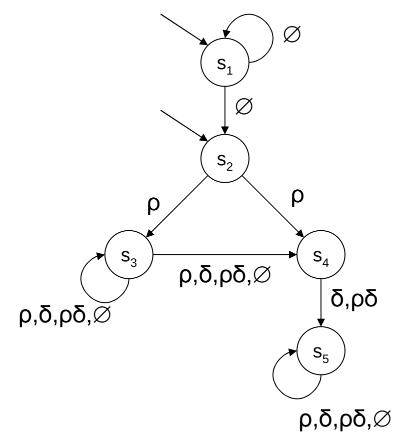
The algorithm works with a set of unprocessed states for which successors have to be generated. These states, and their additional information in the form of 'next requirements' (the formulas that have to hold in all the immediate successors of the state) is maintained in the form of a tuple $(\mathbf{s}, \mathbf{x})$ and stored in the *unp* list. At the start of the algorithm this set of unprocessed states contains only the states that are generated from saturating the formula φ (which will become the start states of the automaton). For each of the states $(\mathbf{s}, \mathbf{x})$ in *unp*, the following is done: 1) the state is added to the automaton ($S := S \cup \{\mathbf{s}\}$); 2) the 'next requirements' $\mathbf{x}$ of the state are saturated to create the set of next states $(\mathbf{s}', \mathbf{x}')$ that must be linked to $\mathbf{s}$, these links make sure that the 'next requirements' of $\mathbf{s}$ are represented in the automaton (the label of the transition from $\mathbf{s}$ to $\mathbf{s}'$ is dependent on the propositions that are included in $\mathbf{s}$); 3) all of the next states $(\mathbf{s}', \mathbf{x}')$ are added to the unprocessed list if a) $\mathbf{s}'$ is not yet a state of the automaton, and b) $(\mathbf{s}', \mathbf{x}')$ is not already in the list of unprocessed states. By performing this recursively for all elements in *unp*, an automaton will be generated based on the 'next requirements' of states added to the automaton.

Using this algorithm we can now create an automaton that models all LTL sequences that satisfy the norms of a given domain. The norms, expressed as LTL formulas as presented earlier in this section, form the basis of the formula φ of which the automaton is built. Let us consider, for illustrative purposes, a domain governed by a single deadline $O(\rho < \delta)$ (ρ needs to happen one or more states before δ appears)[13]. If a protocol needs to be created for this domain, we need to extract the landmarks specified by the norms governing the domain. These landmarks, as explained earlier, are the characteristic features of the norms that define the basic structure of protocols that comply with that set of norms. The landmarks are extracted from the norms by creating an automaton (which, in fact, is a general, canonical-like model of the norms, since it represents all LTL sequences satisfying that set of norms) by means of the procedure described above. However, building an automaton on basis of the logical representation of the norms gives a model of the norms that also includes the LTL sequences where (one of the) norms have been violated, since the violation of a norm is just as much a part of the logical representation because of its prescriptive nature (norms express what should be, not what is). Instead, we are more interested in only those LTL sequences where the norms hold but are not violated, since these sequences characterise the patterns that we want to capture in the protocol that we are creating. In case of our example, this means we need to build an automaton for the formula $O(\rho < \delta) \wedge \Box\neg v(\rho, \delta)$[14]. The resulting automaton A_φ, generated by the protocol described above, is shown in figure 5.3 (a transcript of the manual run of the protocol can be found in Appendix B). The alphabet of the automaton is taken as $\Sigma_\varphi = 2^{\{\rho, \delta, v(\rho,\delta)\}}$, the states $s_1, \ldots, s_5 \subseteq 2^{cl(\varphi)}$ (parts of the state labels are given in figure 5.3), the starting states of the automaton are s_1 and s_2, and the acceptance set is defined as $\mathcal{F}_\varphi = \{\{s_4, s_5\}, \{s_2, s_4, s_5\}\}$.

As can be seen, this automaton exactly represents our intuition of a deadline (as expressed in LTL logics). All LTL sequences satisfying a deadline should have a number of states (possibly zero) in which nothing interesting happens (this is

[13]Examples of such norms, a few already given in chapter 4, are, for instance, the obligation to pay for the goods one has won in an auction before leaving, or, in a medical context, the need to certify that a patient is truly (medically) dead, before extracting the organs of the patient for transplantation purposes.

[14]Note that building an automaton for this formula is actually the same as building an automaton for a simplified representation of an obligation that cannot be violated: $\Diamond\delta \wedge [(\neg\delta \wedge \neg\rho \wedge \neg v)$ **until** $(\rho \wedge \neg\delta \wedge \Box\neg v)]$.



$s_1 = \{O(\rho<\delta), \neg\rho\wedge\neg\delta\wedge\neg v(\rho,\delta), \Diamond\delta, \Box\neg v(\rho,\delta)\}$
$s_2 = \{O(\rho<\delta), \rho\wedge\neg\delta\wedge\Box\neg v(\rho,\delta), \Diamond\delta, \Box\neg v(\rho,\delta)\}$
$s_3 = \{\Diamond\delta, \Box\neg v(\rho,\delta)\}$
$s_4 = \{\Diamond\delta, \delta, \Box\neg v(\rho,\delta)\}$
$s_5 = \{\Box\neg v(\rho,\delta)\}$

Figure 5.3: *A Büchi automaton for the formula $O(\rho < \delta)$.*

represented in state s_1). Then a state occurs where ρ holds (state s_2), after which, one or more states later, δ holds (the intermediate states are represented in state s_3, the state where δ occurs is represented in state s_4). After the obligation has been fulfilled (δ has happened, while ρ happened one or more states before δ), an infinite sequence of states occurs where anything can happen as long as $v(\rho, \delta)$ does not happen; this is represented in state s_5 (since no more restrictions are posed on ρ and δ, they can hold in any order at any state after the deadline has been fulfilled). Note that, as required, none of the states satisfies $v(\rho, \delta)$. To fulfil the Büchi acceptance condition the sequences before ρ has happened (i.e., the transition from s_1 to s_1), and the sequence before δ happens (i.e., the transition from s_3 to s_3), can only be of finite length, the only infinite recursion in this automaton is the transition from s_5 to itself.

5.3.2 Extracting the Normative Landmarks

The next step of the process is the extraction of the normative landmark pattern. The idea is that the Büchi automaton that we created through the procedure mentioned above is translated to an equivalent regular expression. Since the Büchi automaton represents all the LTL models that satisfy the norms, it expresses all characteristics that these models share. It is exactly these shared characteristics that we are interested in, as these define the landmarks that we need to construct the protocol. These characteristics are best distinguished in the pattern that is represented by the automaton, this pattern is most clearly distinguishable in a regular expression.

However, since we translate the norms to a generalised Büchi automaton, we first need to translate the automaton to a normal Büchi automaton before we can extract the regular expression that characterises the automaton.

Generalised Büchi Automata and Büchi Automata

As we mentioned before, the difference between generalised Büchi automata and normal Büchi automata lies in the definition of the acceptance condition; normal Büchi automata accept words with an infinite part that intersects a set of acceptance states, where for generalised Büchi automata this infinite part of the word has to intersect a set of sets of acceptance states. The difference makes it possible for generalised Büchi automata to more easily express complex acceptance conditions (as the one used above, where the acceptance condition expresses that every eventuality in the LTL formula, which can be more than one, has to be reached). The set of sets of acceptance states does, however, make it harder to determine the regular expression characterising the automata, therefore we need to translate the generalised Büchi automata that results from the process mentioned above into a normal Büchi automata.

This translation from a generalised Büchi automaton to a normal Büchi automaton shows some similarities to the translation from a nondeterministic automaton to a deterministic automaton (see [Hopcroft et al., 2006; Sipser, 1997] for details about the translation of nondeterministic automata to deterministic automata). This similarity is mostly because the translation from generalised to a normal Büchi automaton is also about reducing nondeterminism, namely the nondeterminism of the acceptance condition. The idea is as follows. We expand the set of states of the generalised Büchi automaton by attaching an index to them. We use as many indices as there are state sets in the acceptance set (i.e., $\{1, \ldots, k\}$ for the acceptance set of a generalised Büchi automaton defined as $\mathcal{F} = \{F_1, \ldots, F_k\}$). These indices will remain unchanged unless a state is visited that is in F_j where j is the current value of the index (this is done modulo k, i.e., the index is reset to 1 if incremented from a state indexed by k).

Definition 5.7 *Given a generalised Büchi automaton $A = (\Sigma, S, \Delta, S_0, \mathcal{F})$, where $\mathcal{F} = \{F_1, \ldots, F_k\}$, the equivalent Büchi automaton $A' = (\Sigma, S', \Delta', S'_0, F')$ defined as follows.*

- $S' = S \times \{1, \ldots, k\}$.
- $S'_0 = S_0 \times \{1\}$.
- *Δ' is defined by $(t, i) \in \Delta'((s, j), a)$ if*

$$t \in \Delta(s, a) \text{ and } \begin{cases} i = j & \text{if } s \notin F_j, \\ i = (j \bmod k) + 1 & \text{if } s \in F_j. \end{cases}$$

- $F' = F_1 \times \{1\}$.

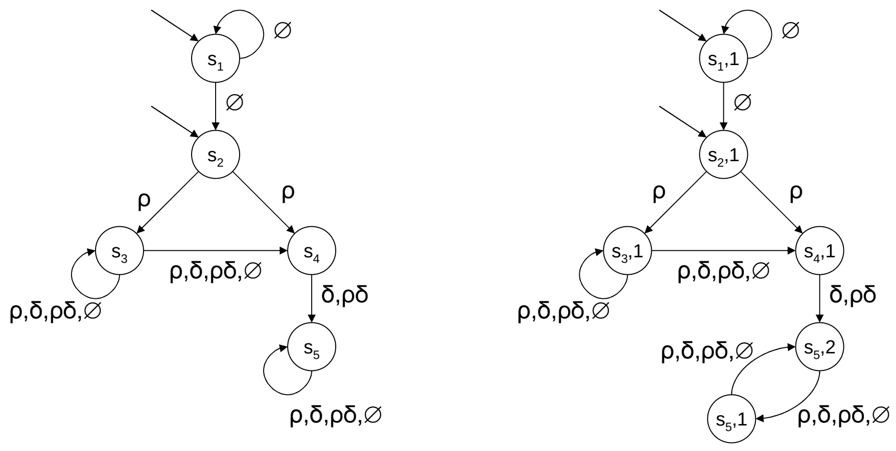


Figure 5.4: *The generalised Büchi automaton (left) and the equivalent normal Büchi automaton (right) for $O(\rho < \delta)$.*

Proposition 5.8 *The Büchi automaton A' accepts exactly the same language as the generalised Büchi automaton A.*

Basically, if one repeatedly cycles through all indices all sets in $\mathcal{F}$ are visited infinitely often, and cycling through the indices is necessary for F' to be reached. Conversely, if it is possible to visit all sets in $\mathcal{F}$ infinitely often in A, it is possible to do so in the order $F_1, F_2, \ldots, F_k$ and hence to infinitely often go through F' in A'.

Given definition 5.7 we can translate the generalised Büchi automaton presented in figure 5.3 to a normal Büchi automaton. The Büchi automaton presented in figure 5.4 is the result of this process. The alphabet of the normal Büchi automaton is the same as that of the generalised automaton, the starting states are states $(s_1, 1)$ and $(s_2, 1)$, and the acceptance set is defined as $F' = \{(s_4, 1), (s_5, 1)\}$. Note that not much has changed from the generalised automaton to the normal automaton, since most newly created states have been cut due to being unreachable.

Büchi Recognisable languages

After translating the norms to a generalised Büchi automaton and converting that automaton to a normal Büchi automaton, we can extract a regular expression characterising the language expressed by the automaton. The idea is that the main characteristics, the basic landmark structure, obtained from the norms that are represented in the Büchi automaton can be easily extracted through this regular expression.

Regular expressions are an algebraic description of a language, capable of defining exactly the same type of languages that are expressible with finite automata; namely regular languages, [Hopcroft et al., 2006; Sipser, 1997]. The advantage of using a regular expression is that it is a clear and compact representation of a language. Regular expressions consist of constants and operators that denote sets of words and operations over these sets, respectively. Given a finite alphabet Σ a regular expression is defined as follows:

- $\{\}, \epsilon, a$ are regular expressions (for all $a \in \Sigma$)[15];
- if R and S are regular expressions, then so are $R.S$ (concatenation), $R + S$ (choice) and R^* (Kleene Star)[16].

Due to the fact that both regular expressions as well as finite automata, though fundamentally different approaches, can be used to express the same type of languages, we can translate an automaton to a regular expression and vice versa.

Büchi automata accept a specific type of regular languages, namely ω-regular languages (or ω-languages, see [Thomas, 1990; Staiger, 1997]). The major difference between ω-languages and regular languages is, like the difference between Büchi automata and normal finite automata, that ω-languages are composed of infinite sequences (ω-words), where regular languages only contain finite words (the sequences are of finite length). Therefore, given a finite alphabet Σ, we can differentiate between the sets of finite words Σ^* and infinite words Σ^ω, and, moreover, it holds for any ω-language that $L_\omega \subseteq \Sigma^\omega$ and for any regular language that $L \subseteq \Sigma^*$. A language recognised by a Büchi automaton A is then defined as $L_\omega(A) = \{w \in \Sigma^\omega \mid A \textit{ accepts } w\}$[17]. The following property of Büchi automata, taken from [Büchi, 1962] then expresses exactly the language recognised by a Büchi automaton A.

Theorem 5.9 *An ω-language $L_\omega \subseteq \Sigma^\omega$ is Büchi recognisable iff L_ω is a finite union of sets $U.V^\omega$ where $U, V \subseteq \Sigma^*$ are regular sets of finite words.*

For a closer analysis of Büchi recognisable ω-languages we use the following notations, given a fixed Büchi automaton $A = (\Sigma, S, \Delta, S_0, F)$: if $w = a_0 \dots a_{n-1}$ is a finite word over alphabet Σ, we write $s \xrightarrow{w} s'$ if there is a state sequence $s_0, \dots, s_n$ such that $s_0 = s$, $(s_i, a_i, s_{i+1}) \in \Delta$ for $i < n$, and $s_n = s'$. We can then define $W_{ss'} = \{w \in \Sigma^* \mid s \xrightarrow{w} s'\}$, i.e., the set of all finite words that can be created by state sequences from s to s'. Each of these finitely many languages $W_{ss'}$ is regular and by definition of Büchi acceptance, we can then define the

[15]Note that we need $\{\}$ for representing the empty set, where most work on regular expressions (also) use $\varnothing$, since in our approach $\varnothing$ is an element of Σ representing the LTL state where nothing holds.

[16]We use R^+ as an abbreviation for $R.R^*$.

[17]A definition of what a Büchi automaton accepts is given on page 111.

ω-language recognised by A as

$$L_\omega(A) = \bigcup_{s \in F} \bigcup_{s_0 \in S_0} W_{s_0 s}.(W_{ss})^\omega.$$

A representation of an ω-language in the form $L_\omega = \bigcup_{i=1}^n U_i.V_i^\omega$, where U_i, V_i are given by regular expressions, is called an ω-regular expression.

For the automaton presented in figure 5.4 this means the following (we assume a state labelling where $t_1 = (s_1, 1), t_2 = (s_2, 1), t_3 = (s_3, 1), t_4 = (s_4, 1), t_5 = (s_5, 1), t_6 = (s_5, 2)$). The language accepted by the automaton A_φ presented in that figure comes down to

$$L_\omega(A_\varphi) = W_{t_1 t_5}.(W_{t_5 t_5})^\omega \cup W_{t_2 t_5}.(W_{t_5 t_5})^\omega$$

in accordance with theorem 5.9 and the formula presented above (the languages $W_{t_1 t_4}.(W_{t_4 t_4})^\omega$ and $W_{t_2 t_4}.(W_{t_4 t_4})^\omega$ are cut, due to $W_{t_4 t_4}$ being empty; there exist no state sequences from t_4 to t_4). The ω-regular expression representing this language is built from the regular expressions for the partial languages used in $L_\omega(A_\varphi)$:

$$W_{t_1 t_5} = \varnothing^+.\rho.all^*.(\delta + \rho\delta).all^+$$

$$W_{t_2 t_5} = \rho.all^*(\delta + \rho\delta).all^+$$

$$W_{t_5 t_5} = (all.all)^*$$

where we use all to denote $(\varnothing + \rho + \delta + \rho\delta)$, basically expressing that everything can possibly hold with the exception of $v(\rho, \delta)$. Combining these expressions we get the following ω-regular expression denoting the language of the automaton of figure 5.4:

$$L_\omega(A_\varphi) = \varnothing^+.\rho.all^*.(\delta + \rho\delta).all^+.(all.all)^\omega \cup \rho.all^*(\delta + \rho\delta).all^+.(all.all)^\omega.$$

The two expressions can be taken together by using $\varnothing^*$ (since $\varnothing^* = \varnothing^0 + \varnothing^+$, where for $\varnothing^0$ expresses a start at state t_2 and $\varnothing^+$ the start at state t_1). Moreover, due to infinite iteration of the state sequence from t_5 to itself (by means of t_6), this can be further simplified to

$$L_\omega(A_\varphi) = \varnothing^*.\rho.all^*.(\delta + \rho\delta).all^\omega.$$

As can be easily seen from this ω-regular expression, the points of interest of every LTL sequence satisfying φ are the state where ρ holds and the state where either δ or $\rho\delta$ holds (the former being the state where $\delta \wedge \neg\rho \wedge \neg v(\rho, \delta)$ holds, the latter where $\delta \wedge \rho \wedge \neg v(\rho, \delta)$ holds). As seen in the expression, confirming the intuition about deadlines, all LTL sequences satisfying $O(\rho < \delta)$ have a state where ρ holds (possibly preceded by a number of states where neither ρ nor δ hold), which always happens before a state occurs where δ holds (the ω-regular expression allows a number of states, possibly zero, between the occurrence of ρ

and δ, in which ρ may occur as well). The fact that the state where δ should hold is denoted in the expression as $\delta + \rho\delta$ is mainly because, since the restriction of ρ at least happening before δ has already been met, at this point it does not really matter whether ρ holds or not (basically, the transition label $\delta + \rho\delta$ expresses no information concerning ρ or $\neg\rho$, but expresses that *at least* δ holds).

Given that we can deduce from the label $\delta + \rho\delta$ that at least δ holds, we can simplify the regular expression to the following landmark pattern

$$\mathfrak{L}_\varphi = \langle \{\rho, \delta\}, \varnothing, \{(\rho, \delta)\} \rangle$$

This is the normative landmark pattern extracted from the norms of the domain (just $O(\rho < \delta)$ in this case). This landmark pattern is the basis of the landmark pattern that we construct to create protocols. The landmark pattern will now be extended with additional landmarks to denote domain-specific information and information to increase the efficiency and feasibility of the pattern.

5.3.3 Strengthening the Pattern

In subsection 5.2.1 we presented the idea of using landmarks to bridge the gap between norms and practice (in this particular case, the norms and the protocols). Part of this idea to create the bridge, taken from the relation between norms and practice in the real world, was to extend the landmarks extracted from the norms with additional information. We have shown above how the basic landmark pattern, the normative landmark pattern, can be extracted semi-automatically from the norms. Similar to what we explained in subsection 5.2.1, we will use this 'skeleton' landmark pattern and extend it with additional landmarks (and orderings between landmarks) to create a landmark pattern that also incorporates, next to the restrictions on the protocol given by the norms, information about efficiency and feasibility that is normally not included in the normative specification of the domain, but rather taken from the practice itself.

Basically we extend the normative landmark pattern with additional landmarks (or orderings over (existing) landmarks) to add information about efficiency and feasibility to the normative landmark pattern. This would mean that the landmark pattern for our example domain, $\mathfrak{L}_\varphi = \langle \{\rho, \delta\}, \varnothing, \{(\rho, \delta)\} \rangle$, is extended with additional landmarks and orderings. The additional landmarks are, in fact, filling in the 'gaps' between the normative landmarks in the ω-regular expression of this normative domain, $L_\omega(A) = \varnothing^*.\rho.all^*.(\delta + \rho\delta).all^\omega$. We will show that adding these additional landmarks, after re-running the procedure mentioned above, creates a new ω-regular expression that represents a new language L'_ω that is accepted by the newly generated automaton A'_φ. To achieve this we have to extend our original LTL-expression with the additional information taken from the practice.

Let us assume that a landmark γ can be added to the normative landmark pattern, between ρ and δ, to increase efficiency. The desired landmark pattern

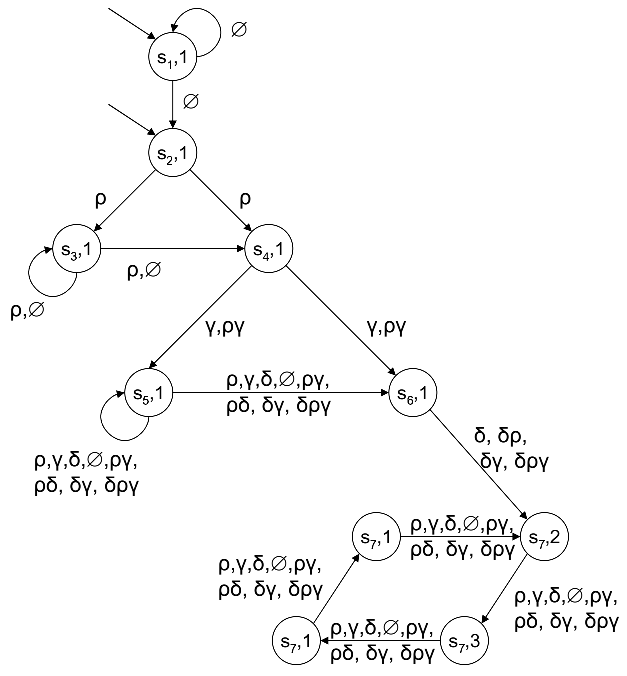


Figure 5.5: A'_φ*: extension of automaton* A_φ *with additional procedural information.*

resulting from this addition would be

$$\mathfrak{L}'_\varphi = \langle \{\rho, \delta, \gamma\}, \varnothing, \{(\rho, \gamma), (\gamma, \delta), (\rho, \delta)\}\rangle.$$

To achieve this we extend our original LTL-expression $O(\rho < \delta)$ with additional LTL-formulas to express these temporal orderings:

$$O(\rho < \delta) \wedge \neg\gamma \textbf{ until } \rho \wedge \neg\delta \textbf{ until } \gamma$$

After re-running the procedure mentioned above, we create a new automaton for this extended LTL-formula. Note that if the added landmark would introduce non-norm-compliant information into the landmark pattern, no models would exist for the LTL-formula, and, subsequently, no automaton would be able to be built from it. The (non-generalised) automaton for the extended LTL-expression is shown in figure 5.5.

We now follow the same process as described above. We translate the generalised automaton to a normal Büchi automaton (which is shown in figure 5.5, and extract the ω-regular expression defining the language accepted by the automaton. In this example, the new ω-regular expression is the following (we use *all′*

for denoting $\rho + \gamma + \delta + \varnothing + \rho\delta + \rho\gamma + \gamma\delta + \rho\gamma\delta$):

$$\varnothing^*.\rho.(\rho + \varnothing)^*.(\gamma + \gamma\rho).(all')^*.(\delta + \delta\rho + \delta\gamma + \delta\rho\gamma).(all')^\omega$$

Extracting from this ω-regular expression the landmark pattern, as described above, results in the desired pattern expressing that ρ should be done first, then γ, and then δ:

$$\mathfrak{L}'_\varphi = \langle\{\rho, \delta, \gamma\}, \varnothing, \{(\rho, \gamma), (\gamma, \delta), (\rho, \delta)\}\rangle.$$

Given that the procedure to create an automaton from LTL is technically quite simple (and easily implemented), this process, while ensuring the norm compliance of the pattern, is not too resource dependent to be run several times. However, remember that, next to the restriction that the extended landmark pattern should remain norm-compliant (which is guaranteed by this process), the extended landmark pattern must satisfy the reachability and capability constraints specified in subsection 5.2.1.

5.3.4 From Landmarks to Protocols

Given that the landmark pattern expresses states that should be achieved, it can be viewed as a collection of concrete goals and the order in which these goals must be achieved. A translation from a landmark pattern to a basic, prototypical protocol is then achieved by means of use of *seeing to it that* operators (*stit*) [Belnap and Perloff, 1988]. While the *stit* operator ignores the means by which an agent will bring about a state of affairs, it does provide the link to make states (the landmarks) into procedural goals. It is then possible to create a protocol given this specification of goals (while retaining the order in which they should be achieved) by linking these goals to the capabilities of agents via a planning algorithm, e.g., like STRIPS [Fikes and Nilsson, 1971].

Let us illustrate this by means of our example, where it means that the landmark pattern $\mathfrak{L}'_\varphi = \langle\{\rho, \delta, \gamma\}, \varnothing, \{(\rho, \gamma), (\gamma, \delta), (\rho, \delta)\}\rangle$ is translated to an ordering of goals, represented as a sequence of (abstract) actions: $(stit\,\rho)\,;\,(stit\,\gamma)\,;\,(stit\,\delta)$, thus expressing that it should be the case that ρ is achieved first, then γ and finally δ.[18] The capabilities of the agents in the domain can then be used to expand this action sequence composed of abstract actions (which abstracts from the means necessary to achieve the expressed goals) to a full protocol (again, expressed as a sequence of actions). For this example, let us assume that the agents have the following capabilities (we use a 'STRIPS-ready' representation of the agent's capabilities, by expressing the necessary preconditions and postconditions of the actions. A definition of an agent's capabilities in such way is found in, for example,

[18]Actually, due to δ being the deadline expressed in the norm, it is not necessarily a goal, but can just as well be an event that just happens. For this example we assume that δ is the goal of the task as a whole, and the norm is just describing that something must be done before that task can be completed.

the 3APL agent programming language, [Dastani et al., 2005]):

$$Op(\mathsf{Action} : action_1,\ \mathsf{Effect} : \rho)$$

$$Op(\mathsf{Action} : action_2,\ \mathsf{Precond} : \rho,\ \mathsf{Effect} : \eta)$$
$$Op(\mathsf{Action} : action_3,\ \mathsf{Precond} : \eta,\ \mathsf{Effect} : \gamma)$$
$$Op(\mathsf{Action} : action_4,\ \mathsf{Precond} : \gamma,\ \mathsf{Effect} : \delta)$$

Using the agent's capabilities, we can use the planning algorithm to expand the abstract protocol $(stit\,\rho)\,;\,(stit\,\gamma)\,;\,(stit\,\delta)$ to a full protocol for the domain, in this case $action_1\,;\,action_2\,;\,action_3\,;\,action_4$.

As hinted in subsection 5.2.1, it might be possible that a landmark expresses a state that is not reachable by a single agent alone; a cooperation of agents might be needed to achieve (parts of) the landmark pattern. In this case, methods for designing interaction patterns between agents, such as the ones in OperA, [Dignum, 2004], can be applied to create the necessary interaction protocols for reaching those complex goals.

5.4 An Example

Let us look at an example to show the entire process described in the previous sections. In the previous section we have shown how the translation from norms to protocols is done via an intermediate level of landmarks. This process starts with the LTL representation of the norms, and ends with a protocol (for a specific task) that is compliant to these norms.

The example we use here is based in the domain of organ transplantation. The task at hand, that should be achieved by the protocol, is to assign organs that have just become available (a suitable donor has been found, i.e., a patient who made a statement to permit post-mortem liver transplantation has died). This task, *assign_organ*, is the goal of the protocol that has to be achieved while keeping in mind the restrictions given by the norms of the domain. We assume that there exists one norm for this task (from the Dutch law on organ transplantations):

> "Before an organ is removed, death is certified by a professional doctor in accordance with the latest medically valid methods and criteria for determining brain death".

We represent this norm as the following (LTL) formula:

$$O(certify\_death < remove\_organ)$$

Naturally, there exist a precedence ordering between the *remove_organ* state and the *assign_organ* state (one cannot assign organs before extraction). Since we use

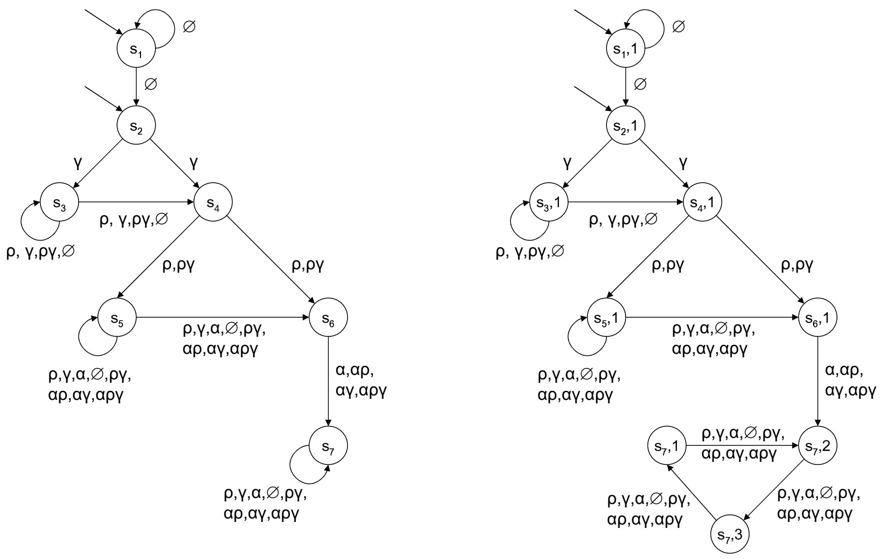


Figure 5.6: *Example Büchi automaton (left) and generalised Büchi automaton (right).*

the *assign_organ* state as the ultimate goal state, we need to model this ordering as well:

$$\neg assign\_organ \ \textbf{until} \ remove\_organ \wedge \Diamond assign\_organ$$

This formula expresses that $\neg assign\_organ$ necessarily holds up to the point at which *remove_organ* holds, after which, at some moment, *assign_organ* will hold. Combined with the norm specified above, this gives us the LTL formula ψ that restricts the domain, and needs to be converted to a landmark pattern:

$$\begin{aligned}\psi \equiv\ & O(certify\_death < remove\_organ) \wedge \\ & \neg assign\_organ \ \textbf{until} \ remove\_organ \wedge \Diamond assign\_organ\end{aligned}$$

The Büchi automaton A_ψ resulting from the procedure described in definition 5.6 is shown in figure 5.6 (we abbreviate *remove_organ* to ρ, *assign_organ* to α and *certify_death* to γ). By means of definition 5.7 we translate this automaton to a generalised Büchi automaton (the automaton shown on the right in figure 5.6). The most important elements of the state labels are the following:

$$
\begin{aligned}
s_1 &= \{O(\gamma < \rho), \neg\alpha \textbf{ until } \rho, \Diamond\rho, \Diamond\alpha, \neg\gamma \wedge \neg\rho \wedge \neg v(\gamma,\rho), \neg\alpha\}\\
s_2 &= \{O(\gamma < \rho, \neg\alpha \textbf{ until } \rho, \Diamond\rho, \Diamond\alpha, \gamma \wedge \neg\rho \wedge \Box\neg v(\gamma,\rho), \neg\alpha\}\\
s_3 &= \{\Diamond\rho, \Diamond\alpha, \Box\neg v(\gamma,\rho), \neg\alpha\}\\
s_4 &= \{\Diamond\rho, \Diamond\alpha, \Box\neg v(\gamma,\rho), \rho, \neg\alpha\}\\
s_5 &= \{\Diamond\alpha, \Box\neg v(\gamma,\rho)\}\\
s_6 &= \{\Diamond\alpha, \Box\neg v(\gamma,\rho), \alpha\}\\
s_7 &= \{\Box\neg v(\gamma,\rho)\}
\end{aligned}
$$

The acceptance set is given as $\mathscr{F}_\psi = \{\{s_6, s_7\}, \{s_4, s_5, s_6, s_7\}, \{s_2, s_4, s_5, s_6, s_7\}\}$, and the acceptance set of the generalised automaton is $F_\psi = \{(s_6, 1), (s_7, 1)\}$.

Using the generalised automaton of figure 5.6 we create an ω-regular expression to describe the language accepted by the automaton. This ω-regular expression captures all the important features of the LTL structures of ψ that are represented in A_ψ. We abbreviate $(\rho + \gamma + \alpha + \varnothing + \rho\gamma + \alpha\rho + \alpha\gamma + \alpha\rho\gamma)$ to *all* (it denotes that everything but $v(\gamma, \rho)$ can hold at that moment).[19]

$$L_\omega(A_\psi) = \varnothing^*.\gamma.(\rho + \gamma + \rho\gamma + \varnothing)^*.(\rho + \rho\gamma).all^*.(\alpha + \alpha\rho + \alpha\gamma + \alpha\gamma\rho).all^\omega$$

From this ω-regular expression we can derive the following (normative) landmark pattern (remember that $\rho + \gamma\rho$ denotes that *at least* ρ should hold, and that $\alpha + \alpha\rho + \alpha\gamma + \alpha\gamma\rho$ denotes that *at least* α holds):

$$\mathfrak{L}'_\psi = \langle\{\gamma, \rho, \alpha\}, \varnothing, \{(\gamma, \rho), (\rho, \alpha)\}\rangle$$

Using knowledge from the practice we strengthen the landmark pattern with additional landmarks. In this case, we use the knowledge that before the assignment the compatibility between the organ and the donor must be checked, e.g., to see if the blood-type of the donor and the recipient match. To increase the efficiency of the assigning, we can check the blood of the donor when the death has been certified (*check_bloodtype*, shortened as β), giving us the following landmark pattern:

$$\mathfrak{L}_\psi = \langle\{\gamma, \rho, \alpha, \beta\}, \varnothing, \{(\gamma, \rho), (\rho, \alpha), (\gamma, \beta), (\beta, \rho)\}\rangle$$

The last steps of the process is converting the landmark pattern to a basic, prototypical protocol, using *stit* operators: $stit(\gamma); stit(\beta); stit(\rho); stit(\alpha)$, which

[19]Note that the expression allows from multiple occurrences of γ, ρ and α (although they need to occur in a particular order), i.e., words like $\rho.\gamma\rho.\alpha\rho.\alpha^\omega$ are accepted by the automaton, while, in practice, only one occurrence of each of these states is more likely; e.g., after the death of the patient has been certified, it will not have to be done again (for that same patient). Basically, the practice that we are trying to model is best described by the ω-expression: $\varnothing^*.\gamma.\varnothing^*.\rho.\varnothing^*.\alpha.\varnothing^\omega$. This, however, is not a weakness of the technique we are describing here, but merely a weakness of the LTL representation that we used. Since we have not specified in our LTL representation that the states will only happen once, they *can* happen multiple times. The LTL representation only models the restrictions on the ordering of the state occurrences.

is then translated, by using the capabilities of the agents, to a protocol:

> *execute_brain_death_protocol* ; *take_blood_sample* ;
> *test_blood*(*bloodtype*) ; *start_operation* ; *remove_organ*(*liver*) ;
> *assign_organ*(*patient*, *bloodtype*) ; *operate*(*patient*)

5.5 Complex Landmark Patterns: Multi-Norm and Multi-Agent

The examples we have shown above are all based around a single distinctive norm. In the case of the example presented in section 5.4 we used a single norm and an ordering restriction, but the complexity of the whole landmark pattern was still quite low. However, this does not mean that more complex situations (for instance, multiple 'conflicting' norms or domains for multiple agents) cannot be modelled. In this section we show how the procedure described above can be used in domains that are described by multiple norms, or when it concerns multiple agents that each have their own norms (next to the general norms governing the domain).

Due to the fact that the translation from LTL to automata presented above can handle any kind of LTL formula (as long as the formula is expressed in negated normal form), more complex combinations of norms can be modelled and used for the design of protocols. In particular, combinations of norms that restrict each other, such as an obligation to ρ before δ and a prohibition that ρ may not occur more than a few states before δ can be expressed and handled. Basically, any of the complex norm expressions from subsection 4.2.1 can be used and combined to an LTL formula that can be translated to an automaton by means of the algorithm given in definition 5.6. For instance, in figure 5.7 an automaton is shown that is generated for the combination of two distinct deadlines $O(\rho < \delta)$ and $O(\gamma < \eta)$ (instead of cluttering the picture with the labels of the transitions, we have chosen to label the states of the automata with the propositions that hold in that state). As can be seen from this example, the size of the automaton can grow rapidly as more and more norms are added to the domain, and automata with multiple norms, though easy to generate due to the algorithm of definition 5.6, can be hard to draw and comprehend.

To alleviate some of the complexity of the automata generated from the norms, heuristics can be applied to reduce the complexity of the LTL formula that is being used to build the automaton. We have already indicated that the automaton build from

$$\begin{aligned} O(\rho < \delta) \Leftrightarrow \Diamond\delta \wedge \Big[&(\neg\delta \wedge \neg\rho \wedge \neg v(\rho,\delta)) \textbf{ until} \\ &((\rho \wedge \neg\delta \wedge \Box\neg v(\rho,\delta)) \vee (\neg\rho \wedge \delta \wedge \bigcirc v(\rho,\delta)))\Big] \wedge \Box\neg v(\rho,\delta) \end{aligned}$$

is equivalent to the automaton build from

$$O(\rho < \delta) \Leftrightarrow \Diamond\delta \wedge \Big[(\neg\delta \wedge \neg\rho \wedge \neg v(\rho,\delta)) \textbf{ until } (\rho \wedge \neg\delta \wedge \Box\neg v(\rho,\delta))\Big]$$

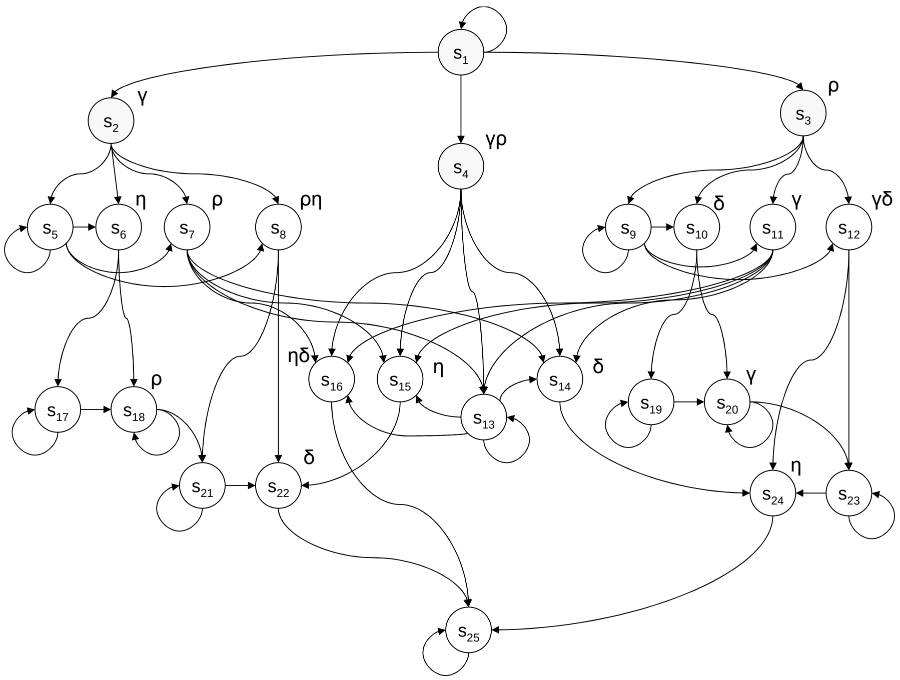


Figure 5.7: *Büchi Automaton for the formula* $O(\rho < \delta) \wedge O(\gamma < \eta)$.

while building the second automaton is less complex. Similar heuristics can be applied, for instance, for the building of an automaton for $O(\rho < \delta) \wedge O(\rho < \gamma)$ (ρ has to appear before both δ and γ appear). In this case, building an automaton for the deadline that occurs the earliest is enough to be compliant to both norms. In case of the automaton of figure 5.7, the process can be simplified by adding additional information about the order in which the deadlines δ and η normally appear (in fact, the formula represented in the automaton of figure 5.7 is rather unusual, since most domains have an (implicit) ordering between the deadlines used).

The point remains, however, that any combination of norms that can be expressed in LTL, and is satisfied by *at least* one LTL sequence, can be used to build an automaton. While the complexity of this automaton and the complexity of the run of the algorithm of definition 5.6 can be alleviated by simplifying the LTL formula, or actually adding additional information to the formula, every set of norms that can be expressed in LTL and is satisfiable can be used to build a landmark pattern. Even modelling time, for instance to express that something should be done within 1 hour after something, can be used to create a landmark pattern, as long as the LTL formula constitutes this aspect of the norm; i.e., the connection between time steps and LTL steps should be made explicit in the modelling of this norm.

Another aspect that can increase the complexity of the process is the design of protocols for multiagent environments. We already mentioned that landmarks that represent goals for a coalition of agents can be used to generate interaction protocols by means of multiagent interaction methodologies such as OperA, [Dignum, 2004]. Using landmark patterns in such environments creates a single interaction protocol which is composed of protocols for each of the individual agents participating in the interaction. This can become even more complex if, instead of just a general set of norms for every agent in the environment, each of the agents is bound by different norms (specified for the different roles that the agents play in the interaction). While the resulting protocol for the entire interaction is guaranteed to be norm-compliant (it was generated from the normative landmark pattern which was extracted from the general norms of the domain), the resulting protocol-parts for each of the agents will have to be tested for norm compliance due to the different norms for each of the different agent roles. A method for checking the norm-compliance of a protocol is presented in chapter 6.

5.6 Chapter Conclusions

In highly restricted domains, such as those discussed in chapter 4, it can be hard for the agents to decide on the right course of action to achieve a certain goal. Protocols can be generated to help agents in frequently done tasks, while making sure that the task is done in an efficient, but more importantly, norm-compliant manner.

The design of protocols for such highly-regulated domains, however, can be hard due to the gap that exists between the abstract, declarative level of the norms and the procedural level of the practice. In this chapter we have shown how protocols can be generated for normative domains by using an intermediate level of landmarks, an idea inspired by the relation between norms, regulations and practice as used in real-world normative systems.

We have shown a technique, based on work done for the research of model-checking by [Vardi and Wolper, 1994], that allows us to create a canonical-like model of the logical representation of the norms. This canonical model, which in fact is a Büchi automaton, contains all the important characteristics of all LTL sequences that satisfy the norms. By means of creating an ω-regular expression to describe the language accepted by the automaton, we are able to easily translate these characteristics to a landmark pattern. This pattern, which is strengthened to increase efficiency and feasibility, is then used to create the protocol.

By using a landmark pattern that is extracted from the norms of the domain we ensure a level of norm compliance in the protocol. If, however, instead of designing protocols by means of this technique, protocols are used that are taken from the real-world practice (and possibly adapted for the agent system) this norm compliance is not 100% certain, and the protocol has to be proven to be norm-compliant. In the next chapter we show how the norm compliance of a protocol can be proven.

Chapter 6

Verifying Norm Compliance

Protocols play an important part in institutions as they are used by agents as a default for doing tasks in the institution. Whether the agents are either unaware of the norms, not capable of reasoning about the norms or just use the protocol as a standard means to reduce the need for elaborate reasoning and planning mechanisms, the protocols lay out the basic norm-compliant manner to achieve the task the agent was given.

We have shown how the protocols for a domain can be designed by using the important steps that are given by the norms: *landmarks*. The landmarks capture the important parts of the norms that are needed for the design of the protocols. Protocols designed by this method provide a basic norm compliance, but for protocols taken from the practice this norm compliance cannot always be guaranteed. The norm compliance of the protocols is, however, an important aspect of the protocol, because of the fact that agents that are not capable of reasoning use them to perform the tasks. If every agent in the institution performs a certain task (by using the protocol that violates norms) this jeopardises the safety and reliability of the system.

In order to guarantee that a protocol is norm-compliant (even for protocols designed by the method of chapter 5), the protocol should be formally verified, which means that it should be checked that no norms are violated by the protocol during its execution. It is important to check this because informal verifications or random checks only make sure that the protocol is norm-compliant in certain states of the world, while no real guarantee can be given about the other states of the world.

In this chapter we present a formal method for checking the norm compliance of protocols. The technique we present is based on an approach used in concurrent programming. To clarify the use of this technique we prove the norm compliance of a real-life protocol used in the domain of organ transplantations.

6.1 Verifying Protocols

In order to guarantee that a protocol is actually norm-compliant, we want to be able to formalise the protocol and applicable norms and prove the norm compliance by deriving that specific propositions hold for that protocol. For the formalisation of protocols we have chosen to regard them as programs, since there is a clear similarity between protocols and programs, mainly because protocols are, like programs, actually nothing more than a set of actions combined with decisions that determine which actions are performed in what situation.

The formalisation of the norms will be done in a linear-time temporal logic, like the one presented in chapter 4. The use of linear time logics in the verification process over more complex temporal logics such as CTL and CTL* is sufficient because the added complexity of branches in time, normally needed for expressing non-determinism, is not required in the domain we are considering. Although the protocols themselves may branch this does not imply the need of a branching time logic, since all branches in the protocol are due to different conditions in the world, and not due to non-determinism. We can, therefore, handle the verification of protocols by using linear-time temporal logic, however, we may need to consider different conditions when checking the protocol, leading to more than one verification process.

Since we consider protocols to be programs and can view the formalised norms as constraints on these programs, the norm compliance of the protocol is reduced to an invariant property holding for this program; that is to say, if we can prove the invariance of non-violation, i.e. no violation is taking place over the entire program run, we can say that the protocol is norm-compliant. This means that, when the protocol is started in a state/world that is violation free, following the protocol to the letter would not make the run include worlds that satisfy a violation. Although the invariance property is enough for checking whether a protocol is norm-compliant, we would rather say a bit more about the protocols we are trying to prove. If, for instance, the protocol is composed of actions that are actually not regulated by the norms in consideration, the protocol is trivially norm-compliant, e.g., being at home all day makes one compliant to the traffic laws all the time, since one is not actually participating in traffic situations. Since such protocols are hardly any interesting for the domain under consideration (`skip`* satisfies almost all violation invariances, though it is not very interesting from an institutional point of view), we want to prove that a protocol is actually goal-oriented. This can be solved by introducing a liveness property that should be satisfied by the protocol.

Clearly, for checking whether a protocol satisfies the properties mentioned above, we need to find out what happens in the states between the start and end of the protocol, which makes traditional (sequential) program verification unfit for this domain. However, a verification method for concurrent programs, such as presented in [Kröger, 1987], combined with a temporal logic, does provide the means necessary for checking whether violations do or do not occur during a protocol run. However, the formalism described in [Kröger, 1987] is, naturally, limited to checking properties and the likes for concurrent programs, not for checking norm compliance; therefore, we had to change and expand the formalism with the means to express norms and violations.

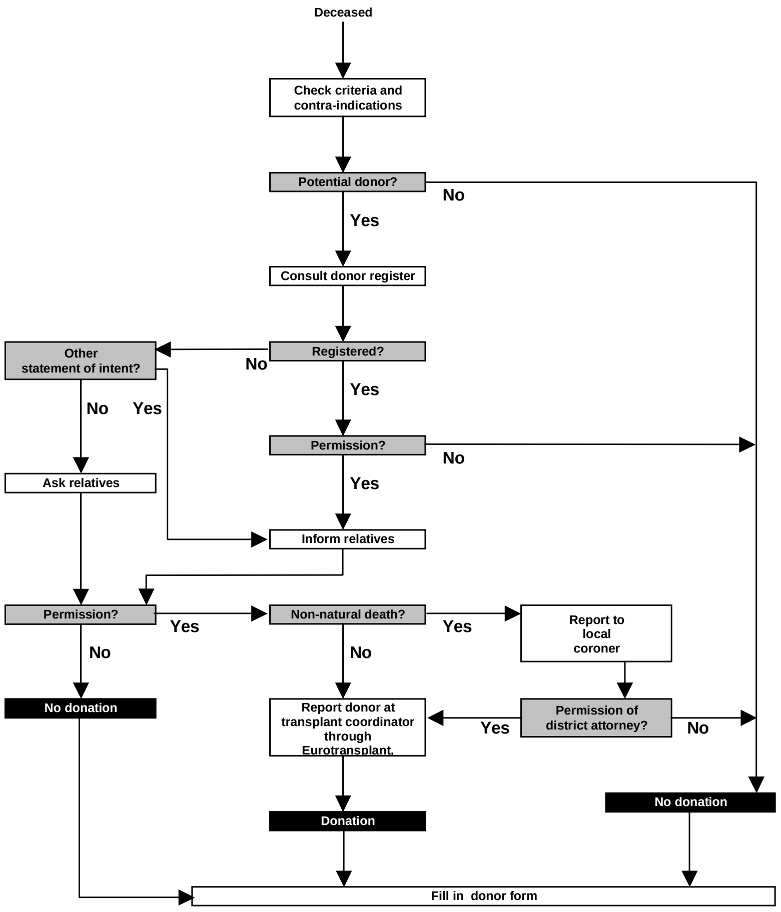


Figure 6.1: *Protocol for organ donation.*

6.2 A Practical Protocol for Organ Donations

The medical domain of organ transplantations is governed by a lot of restricting rules and regulations. Specified by the Dutch law on Organ Transplants, this normative specification creates a highly-regulated domain where it can be hard to decide on the correct order of procedures. To help doctors do what is needed (i.e., legally acquire as many organs for transplant as possible) a model-protocol was created [Dutch Transplant Association, 2006]. This protocol shows the steps that are needed to be taken by a doctor to see if it is allowed to extract the organs of a recently deceased patient. A simplified version of the protocol is included in figure 6.1.

This protocol includes all steps that are needed, specified by law and medical regulations, and claims that no violations will occur if followed precisely. As can be seen in figure 6.1, the protocol is run after the patient has deceased, and specifies that a doctor needs to check whether the patient satisfies all of the listed criteria and none of the contra-indications of becoming a donor. If this first test is successful, the doctor needs to check whether the patient is registered in the Donor Register (this is a special register which contains the approval or disapproval of people to become a postmortal donor), and whether this registration permits the doctor to extract the organs for donation. If no such registration exists, other routes need to be taken to check whether the patient would have approved of donating his organs (this is done by checking for the existence of a statement of intend giving this approval, or, if such a statement does not exist, by consulting the relatives of the patient). Only if the permission for donation can be obtained (through any of these routes), and if the patient did not die from a non-natural death, is the removal of the organs allowed.

In the following we will use this protocol, combined with the norms taken from the law (included in Appendix C), as an example to clarify the process of verifying the norm compliance of a protocol based on the method that will be presented in section 6.3. While formalising this protocol, we encountered some problems, which we will discuss first.

6.2.1 Permissions in a Verification Process

One of the problems we encountered has to do with permissions. Where obligations and prohibitions have a clear intuitive meaning when it comes to searching for violations, i.e. the process of finding out whether norms are violated, permissions have no clear meaning in this context, since, because of their definition, they cannot be violated. Permissions are norms that only define when violations do not occur. Therefore, in a domain where only permissions exist (no obligations or prohibitions), no violations will occur ever, since none of the holding norms can be violated.

Permissions in law are, however, a bit more useful since they are used to reduce the strict restraints that are posed by the obligations and prohibitions. Consider again a domain without any norms, in such a domain the obligations and prohibitions are used to pose restrictions on people/agents in order to define acceptable and unacceptable behaviour and situations. Since some of these restrictions might be too broad, thus restricting too many situations or too many people/agents which should actually be allowed, permissions can be used to counter these effects and reinstate some of the accepted states and actions as being 'legal'.

Some domains are even that restricted that permissions are actually needed to perform actions at all. The domain of organ transplantations is such a domain. This domain is actually a part of a larger and more general domain where regulations and laws restrict people to extract organs from people, since cutting in people and operating on the dead is considered a bad thing (desecration of the dead is still registered as a moral offence in the criminal code). In order to make it possible to extract organs to save lives (which is, of course, considered a good thing) in this domain where everything is considered 'illegal' and therefore prohibited, permissions had to be introduced. These permissions would then define in which situations which actions are actually 'legal', i.e. not raising a violation by breaking the 'default' of not being allowed to violate a corpse.

More interesting is, however, that in such a domain not being permitted to do something would actually mean that your are forbidden to do it, because of this general default, thereby making permission necessary in the checking for violations during protocol runs. For the domain of organ transplantations that we are considering here, this needs to be formalised in order to detect violations that arise because of actions (or states) not explicitly permitted. We model this relation between permissions and prohibitions by means of a technique similar to *negation as failure*, as used in logic programming [Prakken and Sartor, 1997a; van Emden and Kowalski, 1976]; the inability to derive that you are permitted to do α means that you are forbidden to do α:

$$\sim P\,\alpha \rightarrow F\,\alpha$$

Because of the default assumption that anything is prohibited unless specified otherwise, one can say that the legal system modelled by this rule is *restrictive* in nature, thereby characterising the legal system (see [Sergot et al., 1986] for a discussion on the character of legal systems).

Now, since we add the $\sim P\,\alpha \rightarrow F\,\alpha$ rule to our system to model that permissions are exceptions to 'default' prohibitions (where this general prohibition might only follow from the characteristic nature of the law), we get into trouble if we don't assume that permissions follow from obligations (i.e., $O\,\alpha \rightarrow P\,\alpha$). This assumption is an axiom in most deontic systems, but we are reluctant to insert it because we feel that in the real world this might not necessarily hold. It is, however, true that a normative system is supposed to uphold this principle, i.e., normative systems should be designed such that obligations to do α can actually be fulfilled, but this is actually the ideal situation. When designing a normative system (thus, when laws are postulated) it should be taken into account that obligations can be fulfilled. However, it is not necessarily the case that this condition is always met in normative systems (due to mistakes in designing the system). In the case presented in sections 6.4 and 6.5, however, we can safely assume that this assumption has not been violated by the law-maker.

6.2.2 Linking Levels

Norms, as mentioned before, are generally specified by means of vague and ambiguous concepts that not always have **one** clear meaning when translated into concrete situations. Norms, such as the law, are specified in this way to ensure that the norm can handle many different concrete situations, even those situations one did not forsee when designing a law. This vagueness makes norms hard to translate into a formal framework while still able to use the formalisation for checking norm compliance of protocols. Since protocols are concrete procedures, the actions taken in such a protocol are not directly mappable to the abstract actions mentioned in the norms. Although the meaning of the abstract action 'to discriminate' is quite clear to most people, the problem is that normal protocols do not include a *discriminate* action; protocols can, however, include actions that, in certain cases, can be considered to be 'discriminative' of nature.

Of course, one can try and iron out all of these discrepancies between this abstract normative level and the concrete protocol when translating the norms (i.e., translate as many abstract actions/concepts into concrete ones), but by doing so one risks losing valuable information. The latter can happen when a vague norm is misinterpreted or situations arise that were not accounted for when translating the norms. This means that situations might arise when a violation occurs (or is absent) according to the interpreted norms, but not according the original norm. In such cases one will have to check the original norms to see whether the occurrence (or absence) of a violation is correct, and since one cannot easily determine in which cases this should be done, all the original norms are to be checked in all cases to determine the correctness of the (non-)violation occurrences.

Instead we rather use a high-level formalisation of the norms which should provide enough room for the formal representation of the norms to keep their vague and ambiguous nature (like we have done before in chapters 3.2 and 4.2). This representation, however, contains, due to the high-level of abstraction, terms and concepts that cannot be clearly related to terms and concepts used in the concrete protocol. Therefore, in order to check whether certain concrete actions and situations contained in the protocol violate a norm we map these concrete actions and situations to the abstract actions and situations described by the norms. These mappings can be of two different forms.

- Either a concrete atomic action a in the protocol is considered to be an instance of an abstract action α mentioned in the norms.
- Or a sequence of concrete atomic actions $a = a_1; a_2; \ldots; a_n$ is considered an implementation of an abstract (complex) actions α mentioned in the norms.

Note that these mappings are one-way only, that is, a concrete action a in a protocol can be considered to be an instance of an abstract action α mentioned in the norms, but since there are many more actions conceivable that can be considered instances of α, we cannot say that a and α are equivalent. These relations between abstract and concrete actions will be formalised by using a *counts as* operator ($DO_x\, a \leadsto_{viol} DO_x\, \alpha$: meaning *doing a counts as doing α in the case of checking for violations*), as we have introduced earlier in 3.3. In this report we use a simplified version of the *counts as* as defined in [Grossi et al., 2004, 2006b]. Since we are only using the $\leadsto$-operators and the proofs are not dependent on the syntax and semantics of this operator we refrain from giving formal meaning to this operator (interested reader should check the formal aspects of the *counts as* in, for instance, [Jones and Sergot, 1996; Grossi et al., 2004, 2006b]).

An obvious advantage of this approach is that the interpretation of the abstract concepts is made clear in the counts-as. If the norm compliance proof of a certain protocol seems questionable to someone, one could check the counts-as definitions to see which interpretations where used.

6.2.3 Knowledge Representation

Although most concepts, such as 'removing organs' or 'being a doctor', can be represented easily in a logical framework, there are concepts that have a more intangible, context dependent meaning which is hard to formalise. Please note, however, that the representational complexity of a concept used in a certain context is also determined by the use intended for the concept. While 'being a doctor' might be formalised in first-order logic as $doctor(x)$, expressing that element x of the domain has the property of being a doctor, and thus leaving the meaning of being a doctor implicit in specification of the domain, a context might require that one explicitly expresses what it means to be a doctor, e.g., $graduated(x, medicine, university) \wedge specialised\_in(x, surgery) \wedge \ldots \rightarrow doctor(x)$. Although, from a formal point of view, the meaning of $doctor(x)$ is still that element x has the doctor property, the logical formalism now also has the ability to express what it means for elements to have this property.

All this means is that trying to represent a domain when verifying the norm compliance of a protocol is actually the process of 'fleshing out' the meaning of the concepts used in that domain. Without this meaning the verification cannot take place and with the wrong meaning the verification (of a protocol that is actually norm-compliant) could fail. In a sense, the process of verifying norm compliance is actually about defining the logical context of the concepts, and therein very similar to the legal domain where legality/illegality of certain events/actions is determined by the interpretation of the details of the investigation.

Of course these representational choices are debatable, that is, one might not agree with a certain representation for this domain, and therefore it also means that the representation can be regarded as an assumption; i.e., given the representation of the concepts presented in section 6.4.1 we can prove that the protocol of figure 6.1 is norm-compliant.

6.3 Logical Formalism

The basis of our formalisation is a method used for the verification of concurrent programs [Kröger, 1987], which uses a linear-time temporal logic (LTL) to formalise programs and verify properties of programs, like mutual exclusion, deadlock freedom, termination, et cetera. To use this formalism we consider the protocol to be a program, which we can then formalise by means of the constructs that we discuss in subsection 6.3.1. In order to check the norm compliance the norms are formalised into a logical representation, very similar to the one presented in chapter 4[1].

The logic for representing the norms is a typical linear-time logic $\mathscr{L}_{TP}$ for a propositional kernel $\mathscr{L}_P$ with operators $\wedge, \vee, \neg, \rightarrow, \leftrightarrow, \bigcirc$ (next), $\Box$ (always), $\Diamond$ (sometime), $\odot$ (just), $\diamondsuit$ (past), and **until**. The meaning of these operators is given in definition 4.3, with the exception of the past-time operators $\odot$ and $\diamondsuit$ which are introduced below. The *just* operator is for denoting that something held in the previous state, while the *past* operator denotes that something held some time before the current state. The meaning of these operators will be given below, after we have introduced the model semantics for $\mathscr{L}_{TP}$.

For the semantics of the temporal propositional logic $\mathscr{L}_{TP}$ we use a, Kripke-like, model semantics based on possible worlds. The semantics are essentially the same as defined in chapter 4, with a slight twist (e.g., we do not include action assignments in the model, assigning which action is done at what time is now part of the protocol, see subsection 6.3.1 for more details). Moreover, we assume that the set of atomic propositions of $\mathscr{L}_{TP}$ is split into two subsets which we will call the set of global and local propositions, respectively. Unlike local propositions, which inherit their truth value from the *temporal state* of the model, global propositions do not change their truth value over time.

Definition 6.1 (Models of $\mathscr{L}_{TP}$)
A propositional model $\mathcal{M} = \langle \mathscr{A}, v, \mathbf{W} \rangle$ for the temporal propositional logic $\mathscr{L}_{TP}$ consists of:

- *a set of atomic actions $\mathscr{A}$;*
- *a valuation function v to define the truth value of the kernel $\mathscr{L}_P$ of $\mathscr{L}_{TP}$;*
- *an infinite sequence $\mathbf{W} = \{\eta_1, \eta_2, \eta_3, \ldots\}$ of states where every η_i is a local propositional valuation with respect to v (i.e., a truth assignment for every local proposition).*

Using these models we can now define the value of formulas in $\mathscr{L}_{TP}$. We introduce a combined valuation function v_s, derived from the global valuation function v and local valuation η_s, to define the truth value of atomic propositions

[1]Here we only provide a small re-iteration of the logic used, for formal semantics we refer to subsection 4.2.1.

in state s of the model. We can then say that a atomic proposition p is true in a state s of model $\mathcal{M}$ (i.e., $\mathcal{M}, s \vDash p$) if and only if $v_s(p) = \mathbf{true}$. The meaning of this expression, and the semantics of the other formulas of $\mathscr{L}_{TP}$ are given in the following definition:

Definition 6.2 (Semantics of $\mathscr{L}_{TP}$)
For every model $\mathcal{M} = \langle \mathscr{A}, v, \mathbf{W} \rangle$, formulas A and B, and $s \in \mathbb{N}_0$:

$$
\begin{array}{rcl}
\mathcal{M}, s \vDash A & \Leftrightarrow & v_s(A) = \mathbf{true} \quad \textit{for every atomic proposition} \\
 & & \textit{where } v_s(A) = v(A) \textit{ for every global proposition } A \textit{ and} \\
 & & \qquad v_s(A) = \eta_s(A) \textit{ for every local proposition } A \\
\mathcal{M}, s \vDash \neg A & \Leftrightarrow & \mathcal{M}, s \nvDash A \\
\mathcal{M}, s \vDash A \wedge B & \Leftrightarrow & \mathcal{M}, s \vDash A \textit{ and } \mathcal{M}, s \vDash B \\
\mathcal{M}, s \vDash \bigcirc A & \Leftrightarrow & \mathcal{M}, s+1 \vDash A \\
\mathcal{M}, s \vDash \Box A & \Leftrightarrow & \forall t \geq s : \mathcal{M}, t \vDash A \\
\mathcal{M}, s \vDash \odot A & \Leftrightarrow & \mathcal{M}, s-1 \vDash A \quad \textit{for all } s > 0 \textit{ and} \\
 & & \mathbf{false} \textit{ otherwise} \\
\mathcal{M}, s \vDash \Diamond\mkern-11mu\cdot\mkern5mu A & \Leftrightarrow & \exists 0 \leq t \leq s : \mathcal{M}, t \vDash A \\
\mathcal{M}, s \vDash A \textbf{ until } B & \Leftrightarrow & \exists t \geq s : \mathcal{M}, t \vDash B \quad \textit{and} \\
 & & \forall s \leq u < t : \mathcal{M}, u \vDash A
\end{array}
$$

The meaning of the remaining operators (e.g., $\vee, \Diamond$, et cetera) can be derived from the meaning of the operators above as they are introduced as the standard abbreviations (e.g., $\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \neg\psi), \Diamond\varphi \equiv \neg\Box\neg\varphi$, et cetera).

Given this basic linear-time temporal logic we can formalise the norms in a similar manner as done in chapter 4; we introduce the deontic operators that we need for expressing the norms as an Anderson's reduction to temporal formulas (such as given in definitions 4.5, 4.7 and further)[2]. Given by the definitions specified in chapter 4, we use the LTL-reductions of the deontic operators:

[2]The semantics of the DO and DONE operators used in these definitions will become apparent below when we introduce the link between protocols and the logic, see page 148 and further for details.

Definition 6.3 (Deontic Operators)

$$\begin{array}{rcl}
O_x(DO_x\,\alpha \;<\; \delta) & \Leftrightarrow_{def} & \Diamond\delta \wedge \Big[(\neg\delta \wedge \neg viol(x, DO_x\,\alpha, \delta) \;\wedge\; \neg DO_x\,\alpha)\ \textbf{until} \\
 & & ((\neg\delta \wedge DO_x\,\alpha \wedge \Box\neg viol(x, DO_x\,\alpha, \delta)) \vee \\
 & & (\delta \wedge viol(x, DO_x\,\alpha, \delta)))\Big] \\
F_x(DO_x\,\alpha < \delta) & \Leftrightarrow_{def} & \Big(\Diamond\delta \rightarrow \Big[(\neg\delta \wedge \neg DO_x\;\alpha \wedge \neg viol(x, DO_x\;\alpha, \delta))\ \textbf{until} \\
 & & (\delta \wedge \neg DO_x\;\alpha \wedge \Box\neg viol(x, DO_x\;\alpha, \delta)) \;\vee \\
 & & (DO_x\;\alpha \wedge \bigcirc viol(x, DO_x\;\alpha, \delta)))\Big]\Big) \wedge \\
 & & \Big(\Box\neg\delta \rightarrow \Box\Big[DO_x\,\alpha \rightarrow \bigcirc viol(x, DO_x\,\alpha, \delta)\Big]\Big) \\
O_x(DO_x\,\alpha < \delta \mid \sigma) & \Leftrightarrow_{def} & \Box((\sigma \rightarrow O_x(DO_x\,\alpha < \delta))\ \textbf{until}\ (DONE_x\,\alpha \vee \delta)) \\
F_x(DO_x\,\alpha < \delta \mid \sigma) & \Leftrightarrow_{def} & \Box((\sigma \rightarrow F_x(DO_x\,\alpha < \delta))\ \textbf{until}\ \delta) \\
P_x(DO_x\,\alpha \mid \sigma) & \Leftrightarrow_{def} & \Box(\sigma \rightarrow (DO_x\;\alpha \rightarrow \neg viol(x, DO_x\;\alpha))) \\
P_x(DO_x\,\alpha) & \Leftrightarrow_{def} & P_x(DO_x\,\alpha \mid \top)
\end{array}$$

For explanation of these operators we refer to our earlier introduction of these deontics in subsection 4.2.1 on page 68 and beyond.

While trying to formalise the norms used for the example we discuss in sections 6.4 and 6.5 we encountered another kind of norm that needed to be expressed. The norm stated that some action α needed to be done after something had been done (which we model as a proposition δ). Clearly we needed an obligation $O_x(DO_x\,\alpha > \delta)$ expressing that one needs to perform α until δ has passed. But simply expressing this in that manner would mean that one can postpone doing α almost indefinitely, without being in violation. Moreover, since we are trying to see whether a protocol is in violation and protocol runs are defined as infinite sequences of protocol states (see subsection 6.3.1), we cannot handle this sort of expressions in trying to determine whether a violation will ever occur, since that would need checking infinite numbers of states to see that α is never actually performed. To this end we introduce the notion of reaction time, which is a more operational view of the norm in question. Although the law does not say when α should be done after δ, by common sense we understand that it means that α should be done sometime *soon* after δ has occurred. We introduce the operator $\odot^i$ with the following properties: a) $\odot^1\varphi \equiv \odot\varphi$ and b) $\odot^i\varphi \equiv \odot^{i-1} \odot \varphi$; thus, intuitively, defining that φ has taken place i steps in the past.

The idea is then to define $O_x(DO_x\,\alpha > \delta)$ the state in which δ holds to trigger a deadline expressing that α should be done in *r.t.* (the reaction time) steps. The length of this reaction time has to be determined by using the preciseness of the formalisation of the protocol, that is to say, formalising a protocol can introduce many steps between two actions, since many checks need to be made before determining which actions are to be executed.

Definition 6.4 (Temporal Triggered Obligations)

$$O_x(DO_x\,\alpha > \delta) \quad \equiv_{def} \quad \Box(\delta \rightarrow O_x(DO_x\,\alpha < \odot^{r.t.}\delta))$$

Note that we restricted ourselves here to the deontic operators that we needed in the formalisation of the norms used in the example described in section 6.4. Of course other kinds of deontic operators can be expressed as well in a similar manner if needed. When formalising other normative domains, one might need to change some of the definitions given above, i.e. simplify or enhance the used definitions.

6.3.1 Syntax and Semantics of Protocols

As mentioned earlier, we consider protocols to be (a special kind of) programs. In order to be able to express protocols/programs we introduce a formal syntax for the protocols. This syntax, representing the protocol, is then linked to the formal models of the logic in order to make it possible to use the logical formalism, as presented in the previous section, for proving various properties of the protocols, including norm compliance. Here we constrain ourselves to the use of single sequential protocols, although more complex protocols, including parallel and cyclic ones, can be expressed and used with the formalism presented after the necessary expansions.

We formally define the structure of a program as the following:

Definition 6.5 (Formal Protocol)
A protocol Π *is defined over propositional logic* $\mathscr{L}_P$ *and set of atomic actions* $\mathscr{A}$ *as a string of the form*

$$\Pi = \textbf{initial}\ R\,; \Pi_c$$

where Π_c *is a* $\langle$non-cyclic protocol component$\rangle$*, and* R *is a formula of* $\mathscr{L}_P$ *called the initial condition.*

This definition shows that protocols are constructed by means of two components; 1) a set of *initial statements* that are used to initialise global variables at the start of the program-run, and 2) a *sequential program component*. This sequential program component is the formal specification of the protocol and uses atomic actions (including variable assignments) from $\mathscr{A}$, **if-then-else** and **while-do** clauses to specify the protocol. The conditions of these latter two statements are logical formulas (from $\mathscr{L}_P$). For ease of reference all statements in a protocol are labelled, with labels being unique throughout the protocol, i.e., no two labels occurring in a protocol are equal. Definition 6.6 shows the structure of the sequential component.

Definition 6.6 (Non-cyclic Protocol Components)

$\langle$non-cyclic protocol component$\rangle$ ::= $\langle$statement sequence$\rangle$; $\langle$label$\rangle$: **stop**
$\langle$statement sequence$\rangle$ ::= $\langle$statement$\rangle$ | $\langle$statement$\rangle$; $\langle$statement sequence$\rangle$
$\langle$statement$\rangle$ ::= $\langle$label$\rangle$: $\langle$unlabelled statement$\rangle$
$\langle$unlabelled statement$\rangle$ ::= $\langle$element of $\mathscr{A}\rangle$ |
if $\langle$condition$\rangle$ **then** $\langle$statement sequence$\rangle$
else $\langle$statement sequence$\rangle$ **fi** |
if $\langle$condition$\rangle$ **then** $\langle$statement sequence$\rangle$ **fi** |
while $\langle$condition$\rangle$ **do** $\langle$statement sequence$\rangle$ **od**

Protocols are executed in discrete steps. This means that each step the program component executes an *(indivisible) action*, which is either a statement of $\mathscr{A}$ (i.e. an atomic action or variable assignment), a test of the condition in an **if** statement together with the entry into the respective branch, or a test of the condition of a **while** statement together with the respective entry into or exit from the loop. This means, that only one step of the protocol, labelled by a protocol label, is performed per discrete step, i.e. if a protocol has steps labelled $\lambda_1, \ldots, \lambda_n$, it performs only one of those labelled actions/steps (λ_i) each step. Therefore, we can distinguish the protocol steps by means of the protocol labels attached to the step performed, that is to say, we can determine the 'state' of the protocol by referring to the label of the step that the protocol is about to perform. For ease or reference we use λ_0 (start label) to denote the first statement of the component Π_c, λ_e (stop label) as the label of the stop statement, and $\mathscr{M}_\Pi$ as the set of labels occurring in Π.

Now that we have shown the syntax of protocols we determine what the constructs introduced in definition 6.6 actually mean. To formalise this 'operational semantics' of a program Π, we introduce three concepts, for any statement sequence ψ, $\mathscr{M}_\psi$ the labels occurring in ψ and $\mathscr{F}(\mathscr{L}_P)$ being the set of all formulas of $\mathscr{L}_P$;

- entry(ψ) $\in \mathscr{M}_\psi$ for determining the entry label of ψ;
- the set trans(ψ) $\subseteq \mathscr{M}_\psi \times \mathscr{F}(\mathscr{L}_P) \times \mathscr{M}_\psi$ defining the transitions between statements in ψ; and
- the set exits(ψ) $\subseteq \mathscr{M}_\psi \times \mathscr{F}(\mathscr{L}_P)$ defining the exit conditions of ψ.

Using these concepts we can inductively define the meaning of the statements mentioned in definition 6.6 above.

Definition 6.7 (Operational Semantics of Protocol Statements)

1. $\psi \equiv \alpha : \langle$element of $\mathscr{A}\rangle$:
$\quad$ entry(ψ) $= \alpha$,
$\quad$ trans(ψ) $= \emptyset$,
$\quad$ exits(ψ) $= \{(\alpha, \top)\}$.

2. $\psi \equiv \alpha : \textbf{if } B \textbf{ then } \psi_1 \textbf{ else } \psi_2 \textbf{ fi} :$
$\text{entry}(\psi) = \alpha,$
$\text{trans}(\psi) = \text{trans}(\psi_1) \cup \text{trans}(\psi_2) \cup \{(\alpha, B, \text{entry}(\psi_1)), (\alpha, \neg B, \text{entry}(\psi_2))\}$
$\text{exits}(\psi) = \text{exits}(\psi_1) \cup \text{exits}(\psi_2).$

3. $\psi \equiv \alpha : \textbf{if } B \textbf{ then } \psi_1 \textbf{ fi} :$
$\text{entry}(\psi) = \alpha,$
$\text{trans}(\psi) = \text{trans}(\psi_1) \cup \{(\alpha, B, \text{entry}(\psi_1)\},$
$\text{exits}(\psi) = \text{exits}(\psi_1) \cup \{(\alpha, \neg B)\}.$

4. $\psi \equiv \alpha : \textbf{while } B \textbf{ do } \psi_1 \textbf{ od} :$
$\text{entry}(\psi) = \alpha,$
$\text{trans}(\psi) = \text{trans}(\psi_1) \cup \{(\alpha, B, \text{entry}(\psi_1))\} \cup \{(\beta, C, \alpha) | (\beta, C) \in \text{exits}(\psi_1)\},$
$\text{exits}(\psi) = \{(\alpha, \neg B)\}.$

5. $\psi \equiv \alpha : \psi_1; \psi_2$ *(*ψ_1 unlabelled statement, ψ_2 statement sequence*) :*
$\text{entry}(\psi) = \alpha,$
$\text{trans}(\psi) = \text{trans}(\psi_1) \cup \text{trans}(\psi_2) \cup \{(\beta, C, \text{entry}(\psi_2)) | (\beta, C) \in \text{exits}(\psi_1)\},$
$\text{exits}(\psi) = \text{exits}(\psi_2).$

Now, in order to complete the semantics we define the transitions for a protocol part Π_c as mentioned in definitions 6.5 and 6.6, i.e. $\text{trans}(\Pi_c) \subset \mathscr{M}_{\Pi_c} \times \mathscr{F}(\mathscr{L}_P) \times \mathscr{M}_{\Pi_c}$, as:

Definition 6.8 (Operational Semantics of Protocols)

$\Pi_c \equiv \psi; \alpha :$ **stop** *(for a statement sequence* ψ*) :*
$\text{trans}(\Pi_c) = \text{trans}(\psi) \cup \{(\beta, C, \alpha) | (\beta, C) \in \text{exits}(\psi)\}.$

Informally $(\alpha, C, \beta) \in \text{trans}(\Pi_c)$ means that: In Π_c, execution can proceed in one step from α to β if C holds. Therefore, $\text{trans}(\Pi_c)$ expresses all possible steps that can be taken in the protocol-part, and thereby describes all possible paths within the protocol.

Let us explain the semantics a bit more by means of a simple example. Consider the following protocol (which is a simplification of, but of similar structure to the protocol we use in sections 6.4 and 6.5):

```
Πc ≡  $\pi_1$ : if $\varphi_1 \wedge \varphi_2$
           then $\pi_2$ : $\langle action_1 \rangle$
           else  $\pi_3$ : if $\varphi_1 \wedge \neg\varphi_2$
                     then $\pi_4$ : $\langle action_2 \rangle$
                     fi
           fi;
      $\pi_5$ : stop
```

After applying the above definition, and decomposing the protocol into the parts and completing the trans and exits for each step we obtain the following:

$$\text{trans}(\Pi_c) \quad = \quad \text{trans}(\pi_1) \cup \{(\beta, C, \pi_5) \mid (\beta, C) \in \text{exits}(\pi_1)\}$$

$$\begin{array}{rcl}
\text{trans}(\pi_1) & = & \text{trans}(\pi_2) \cup \text{trans}(\pi_3) \cup \\
 & & \{(\pi_1, \varphi_1 \wedge \varphi_2, \pi_2), (\pi_1, \neg(\varphi_1 \wedge \varphi_2), \pi_3)\} \\
\text{exits}(\pi_1) & = & \text{exits}(\pi_2) \cup \text{exits}(\pi_3) \\
\text{trans}(\pi_2) & = & \emptyset \\
\text{exits}(\pi_2) & = & \{(\pi_2, \top)\} \\
\text{trans}(\pi_3) & = & \text{trans}(\pi_4) \cup \{(\pi_3, \varphi_1 \wedge \neg\varphi_2, \pi_4)\} \\
\text{exits}(\pi_3) & = & \text{exits}(\pi_4) \cup \{(\pi_3, \neg(\varphi_1 \wedge \neg\varphi_2))\} \\
\text{trans}(\pi_4) & = & \emptyset \\
\text{exits}(\pi_4) & = & \{(\pi_4, \top)\}
\end{array}$$

Which leads, after combining the trans's and exits's from each step of the protocol-part, to the following transitions for Π_c:

$$\begin{array}{rcl}
\text{trans}(\Pi_c) & = & \{(\pi_1, \varphi_1 \wedge \varphi_2, \pi_2),\ (\pi_1, \neg(\varphi_1 \wedge \varphi_2), \pi_3),\ (\pi_2, \top, \pi_5), \\
 & & (\pi_3, \varphi_1 \wedge \neg\varphi_2, \pi_4),\ (\pi_3, \neg(\varphi_1 \wedge \neg\varphi_2), \pi_5),\ (\pi_4, \top, \pi_5)\}
\end{array}$$

This set of transitions precisely captures the paths occurring in protocol-part Π_c, namely $\pi_1 - \pi_2 - \pi_5$, $\pi_1 - \pi_3 - \pi_4 - \pi_5$ and $\pi_1 - \pi_3 - \pi_5$, and thereby expresses exactly what the protocol means.

Linking Protocols to the Logic

The formalism we just introduced for expressing protocols is an exogenous language, and therefore not a part of our logical framework. However, in order to prove properties and aspects of these protocols we need to connect the protocols to the logical model. To accomplish this we introduce the concepts of protocol states and computations (execution sequences), which we can connect to the states in the model of our logical formalism.

Remember that in section 6.3, at the introduction of the base logic, we assumed a partitioning of the propositions into two different kinds, local propositions and global propositions. This distinction is needed for relating protocols to the logic, where we assume that all propositions whose truth value is changed in the protocol (for example, by means of assignment statements) are local propositions, and all others propositions used in the context of the protocol are global propositions. The idea is then that, while the value of global propositions is still assigned by the valuation function v of the logical model, the logical states $\eta_i \in \mathbf{W}$ are connected to a state of the protocol to assign a value to the local propositions.

Definition 6.9 (Protocol States)

For a program $\Pi = \textbf{initial}\ R\,; \Pi_c$ *over* $\mathscr{L}_P$ *and* $\mathscr{A}$*, a program state of* Π *is a tuple* $\eta = (\mu, \lambda_i)$ *where:*

- μ *is a local propositional valuation with respect to* v *(i.e., a truth assignment for every local proposition); and*
- $\lambda_i \in \mathscr{M}_{\Pi_c}$ *(the label of the action to be executed next).*

If $\lambda_i = \lambda_e$ (i.e., the label of the **stop**-statement of Π_c), we call η the *terminal state*. The part μ of a program state η is just a (partial) propositional interpretation as used in definition 6.1. This is, therefore, together with a global interpretation I the mapping (v^μ) that associates every formula of $\mathscr{L}_{TP}$ with a truth value. A run, or computation, of a protocol Π can then be defined as the sequence of protocol states such that: a) the run starts in the (logical) state that satisfies the initial conditions, b) the transition between states are only those included in trans(Π), and c) no changes are made in the model after the protocol has reached its end-label.

Definition 6.10 (Computations of Π)
An execution *sequence of* Π *(with respect to v) is an infinite sequence of* $W_\Pi = \{\eta_0, \eta_1, \eta_2, \ldots\}$ *of protocol states of* Π *(with respect to v) with the following properties:*

- $\eta_0 = (\mu_0, \lambda_0)$ *and* $v^{\mu_0}(R) = \mathbf{t}$*;*
- *If* $\eta_j = (\mu, \lambda_i)$ *then* $\eta_{j+1} = (\mu', \lambda'_i)$ *and* trans(Π) *contains an element* $(\lambda_i, C, \lambda'_i)$ *and* $v^\mu(C) = \mathbf{true}$. *If, moreover,* λ_i *is the label of a statement not in* $\mathscr{A}$ *then* $\mu' = \mu$*;*
- *If* $\eta_j = (\mu, \lambda_e)$ *then* $\eta_{j+1} = \eta_j$.

Definition 6.10 effectively captures the above mentioned link between the exogenous protocols and the linear temporal logic. It makes sure that the states of the model correspond to the states of a run of the protocol. Note, however, that, since protocols have choice points because of the if-then and while statements, there can be various different models for a protocol (i.e., different models for every different run of the protocol). Likewise, because of changes in the value of global variables, there can be more than one model for a single run (i.e., different models for the same run, but with other instantiations of the global variables).

Also note that, although an execution sequence is defined as an infinite sequence of program states, time itself is actually semi-finite (no program-states before the start of the protocol). This means that protocols cannot use information about previous runs or next runs (unless explicitly modelled). If protocols Π_1 and Π_2 are run one after another (i.e., $\Pi_1; \Pi_2$), Π_1 cannot use information from or about Π_2 and vice versa; the protocols Π_1 and Π_2 are considered as separate runs. This also means that the value of program variables, truth values of propositions and states and information gathered is restricted to the runtime of a protocol. Propositions *can* change their truth values during a protocol run, however, but only because of actions that are taken in the protocol.

As explained earlier, the defined protocol is extrageneous to the logic, and although there is a connection between the protocol and the models of $\mathscr{L}_{TP}$ by means of the definition of protocol states, we need some constructs in the logic to represent the actions and their results. To this end we introduce propositions to denote which action is going to be performed in the state, thereby extending the temporal propositional logic $\mathscr{L}_{TP}$ to the temporal propositional logic of protocols $\mathscr{L}_{TP_\Pi}$:

Definition 6.11 (Action Label Propositions)
The language $\mathscr{L}_{TP_\Pi}$ is obtained from the extension of $\mathscr{L}_{TP}$ with:

- $\mathrm{at}\,\lambda$ *for every* $\lambda \in \mathscr{M}_\Pi$

So every label (λ) of a step in the protocol is now included as an (atomic) formula, intuitively meaning that 'the action λ is executed next'. We use these in the logic to denote the state of the protocol, where $\mathcal{M}, s \models \mathrm{at}\,\lambda$ for some protocol state $\eta_s = (\mu_s, \lambda_s)$ in $\mathcal{M}$, if and only if $\lambda = \lambda_s$.

Next we introduce the abbreviations $DO_x\,\lambda \equiv \mathrm{at}\,\lambda$ and $\mathrm{start}_\Pi \equiv \mathrm{at}\,\lambda_0 \wedge R$ (where λ_0 is the start-label of Π, and R are the initial statements of Π). For ease of reference and understanding we also say that $DONE_d\,\alpha \equiv \odot DO_d\,\alpha$, denoting that α was the label of the previous state (and thus stating that α has just been done). Note that the abstract actions in the norms, as specified in C, are considered part of some external, vague protocol that are linked to the concrete protocol as given by the execution sequence by means of the counts as operator that we introduced in subsection 6.2.2.

6.4 Proving Non-Violation Invariance

As mentioned in section 6.1 we check the protocol on norm compliance by specifying a *safety* property that has to be satisfied by the protocol. This safety property is an invariant, i.e. a formula that should hold during the entire execution of the protocol. The safety property for stating that a protocol should be compliant to the norms is as simple as specifying that non-violation should not change over time (from the start of the protocol on). This can be defined as follows:

Definition 6.12 (Safety Property of Protocols)

$$\mathrm{start}_\Pi \wedge \Box Norms \quad \rightarrow \quad \Box\neg violation$$

Where $\mathrm{start}_\Pi \equiv \mathrm{at}\ \alpha_0$ (*the protocol is at its start label*), $Norms$ being the conjunction of all applicable norms, and $violation \equiv \bigvee_{x,\alpha,\delta} viol(x,\alpha,\delta)$ (*violation is the disjunction of all viol-formulas that occur in Norms*). This safety property of protocols is defined as the global invariance of $\neg violation$ for the protocol Π under the condition that $Norms$ always holds, i.e., if $\Box Norms$ holds upon the start of running Π, then $\neg violation$ will hold in all states of the run.

To prove that a protocol satisfies this property we will verify that no actions in the protocol change the value of violations, that is to say, no violation will occur during the execution of the protocol on account of an action in the protocol; thus proving that non-violation is an invariance of all actions in the protocol. Combined with the assumption that the protocol is started in a violation-free state we can derive the safety property specified above. We can make this assumption because we are not interested in the situations where this assumption does not hold, such as the situation in which the protocol is started when a violation has already occurred, since starting the protocol in such a situation would say nothing about the norm compliance of the protocol, only that it cannot 'repair' the situation it started in. This derivation rule can be defined as:

Theorem 6.13 (Invariance Rule) *The following rule is valid:*

$$\frac{\begin{array}{l}\text{start}_{\Pi} \wedge \Box Norms \rightarrow \neg violation \\ \neg violation \ \mathsf{invof} \ \bar{\mathscr{M}}_{\Pi}\end{array}}{\text{start}_{\Pi} \wedge \Box Norms \rightarrow \Box \neg violation}$$

Where C invof $\alpha \equiv \alpha \wedge C \rightarrow C$ (*C is an invariant of α*) and C invof $\mathscr{M} \equiv C$ invof $\alpha_1 \wedge \ldots \wedge C$ invof α_m (*C is an invariant of every $\alpha \in \mathscr{M}$*), and $\bar{\mathscr{M}}_{\Pi}$ is the set of all labels in the program except for the label of stop, the end-statement. This rule is also very close to the intuition one might have about protocols being norm-compliant, namely if there are no steps in the protocol that violate any norm, the protocol will not violate any of the norms as a whole (if no violation existed when the protocol started running).[3]

6.4.1 Assumptions

Before we can prove the safety property (and later on the liveness property) we need to emphasises our assumptions, which we present in this section.

As explained in section 6.2.3 there have been cases where we had to make representational choices in formalising some of the vague concepts used in the norms. A list of these concepts, introduced as domain knowledge, is the following:

1) $\mathcal{M} \models necessary\_for(consult(d, register), intended(organ\_removal)) \leftrightarrow$
$\Box(\neg DO_d\, consult(d, register) \rightarrow \neg\Diamond DO_{d'}\, remove\_organ(d', p))$
2) $\mathcal{M} \models under\_authority(d, consult(d, register))$
3) $\mathcal{M} \models (\neg registered(p) \wedge \neg other\_statement(z, p)) \rightarrow \neg statement\_of\_intent(w, p)$
4) $\mathcal{M} \models ((know\_statement\_permission(d, p) \vee know\_other\_statement(d, p) \vee$
$know\_relative\_permission(d, p)) \wedge (\neg non\text{-}natural\_death(p) \vee$
$(non\text{-}natural\_death(p) \wedge know\_DA\_permission(d, p)))) \rightarrow \Box available(organ(p))$
5) $\mathcal{M} \models known\_non\text{-}natural\_death(d, p) \rightarrow suspicion(d, p, non\text{-}natural\_death)$

[3]A proof of an invariance rule similar to the one specified in theorem 6.13 can be found in [Kröger, 1987].

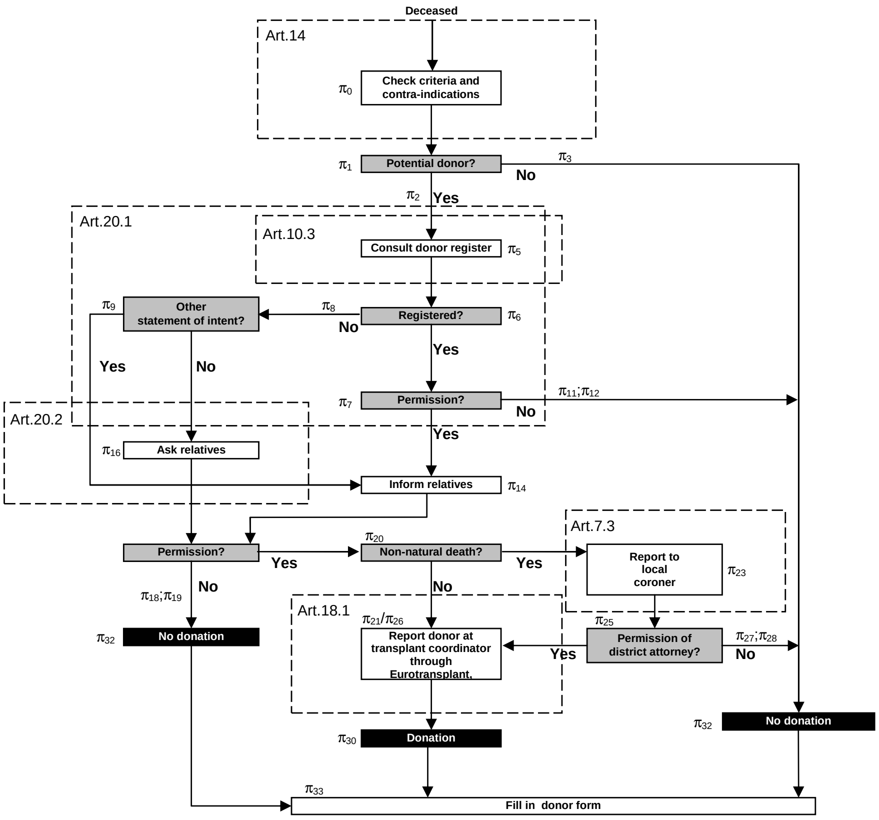


Figure 6.2: *Norms and protocol labels mapped on informal protocol description.*

The first rule states that "consulting the register is necessary for the intended removal of organs" whenever not consulting the register would lead to not doing the organ removal ever. The second rule expresses that (in our model) d (the doctor running the protocol) consulting the register always does this under authority of himself. The third item states that a patient not being registered and not having made another statement of intent can be considered as no statement of intent existing for this patient. The fourth item expresses what it means that an organ becomes available. In the context of this work this means that whenever the patient died of natural causes and the doctor knows that he has the permission to remove the organ (either through a registered consent, another statement of intent, or given by the relatives) the organ is available. In the case that the patient did not die of natural causes, but a permission has been obtained from the district attorney, and the doctor knows he has the permission, the organs are available as well. The last item states that if a doctor knows that the patient has died of non-natural causes, then he will also suspect that the patient died of non-natural causes.

The protocol we are examining is used to 'change' the knowledge of the doctor about whether or not organs may be extracted. For simplification we have made use of propositions to express this knowledge of the doctor. However, some properties of knowledge as expressed in S5 models and epistemic logics in general, see [Meyer and van der Hoek, 1995] for more details on S5 and epistemic logics, are desired for use in our framework, which we had to add by hand. To this end we make the following assumption concerning *all* knowledge-proposition, in order to capture the idea that 'knowledge is truthful':

$$know\_\varphi \rightarrow \varphi$$

$$know\_not\_\varphi \rightarrow \neg\varphi$$

And to capture that knowing φ means that you do not know $\neg\varphi$:

$$(know\_\varphi \wedge know\_not\_\varphi) \rightarrow \bot$$

Another thing needed before we can derive the norm compliance of the protocol formalised in Appendix C is the meaning of the actions that are taken in the protocol. These actions change the world, that is to say, they change the truth value of proposition from one state to another. In table 6.1 below, we specify exactly how this is done.

Intuitively this means that an action (α) results in a specified postcondition (ψ_α) if the corresponding precondition (φ) held in the model at the time the action was performed. Thus, formally if $\mathcal{M} \models \text{at}\,\pi_i \rightsquigarrow \alpha$ then $\mathcal{M}, s \models (\text{at}\,\pi_i \wedge \varphi) \rightarrow \bigcirc\psi_\alpha$.

Note that there are actions specified that do not change the world, i.e., precondition and postcondition are $\top$. These actions are specified in the real-world protocol, and included in the formalised version, but do not change any knowledge of the doctor that is necessary for the safety or liveness proof of the protocol. The actions themselves, however, might be needed for fulfilling obligations.

6.4.2 Invariance Proof

Using the formalism described in section 6.3 we show in this section how the safety property specified earlier is proven for the protocol for obtaining the permission to extract organs for transplantation. This example protocol, introduced in section 6.1 and shown in figures 6.1 and 6.2, as well as the applicable norms, is first formalised (see Appendices A and B). Then, for proving the safety property, i.e. proving that $\neg$*violation* is an invariant of the protocol, we make use of the invariance rule as mentioned in theorem 6.13.

Pre-Condition	*Action*	*Post-Condition*
$criteria(p) \wedge contra\text{-}indication(p)$	⟨check_criteria_&_contra-indications⟩	$know\_criteria(d,p) \wedge know\_contra\text{-}indication(d,p)$
$\neg criteria(p) \wedge contra\text{-}indication(p)$	⟨check_criteria_&_contra-indications⟩	$know\_not\_criteria(d,p) \wedge know\_contra\text{-}indication(d,p)$
$criteria(p) \wedge \neg contra\text{-}indication(p)$	⟨check_criteria_&_contra-indications⟩	$know\_criteria(d,p) \wedge know\_not\_contra\text{-}indication(d,p)$
$\neg criteria(p) \wedge \neg contra\text{-}indication(p)$	⟨check_criteria_&_contra-indications⟩	$know\_not\_criteria(d,p) \wedge know\_not\_contra\text{-}indication(d,p)$
$registered(p)$	⟨consult_donor_register⟩	$know\_registered(d,p)$
$\neg registered(p)$	⟨consult_donor_register⟩	$know\_not\_registered(d,p)$
$statement\_permission(p)$	⟨check_permission⟩	$know\_statement\_permission(d,p)$
$\neg statement\_permission(p)$	⟨check_permission⟩	$know\_not\_statement\_permission(d,p)$
$other\_statement(w,p)$	⟨check_other_statement⟩	$know\_other\_statement(d,p)$
$\neg other\_statement(w,p)$	⟨check_other_statement⟩	$know\_not\_other\_statement(d,p)$
$\top$	⟨inform_relatives⟩	$\top$
$\top$	⟨report_to_transplant_coordinator⟩	$\top$
$\top$	⟨report_to_coroner⟩	$\top$
$DA\_permission(DA,p)$	⟨ask_DA_for_permission⟩	$know\_DA\_permission(d,p)$
$\neg DA\_permission(DA,p)$	⟨ask_DA_for_permission⟩	$know\_not\_DA\_permission(d,p)$
$\top$	⟨**fill_donor_form**⟩	$\top$

Table 6.1: *Pre- and Post-conditions of Protocol Actions.*

In order to prove that the protocol does not violate any of the norms specified in Appendix C, we need to show that all of the obligations that are stated are actually fulfilled. This means that we need to check whether the obliged action is actually performed at the applicable moment, which is before the deadline has passed:

Theorem 6.14 (Derivation Rule) *The following rule is valid:*

$$\frac{\begin{array}{c}\mathcal{M} \models O_x(\alpha < \delta) \\ \mathcal{M} \models \text{at}\,\pi_i \rightsquigarrow \alpha \\ \mathcal{M} \models \text{at}\,\pi_i \rightarrow \neg\Diamond\delta\end{array}}{\mathcal{M} \models \text{at}\,\pi_i \rightarrow \bigcirc\neg viol(x,\alpha,\delta)}$$

Intuitively this theorem states that we can derive that the norm $O_x(\alpha < \delta)$ will not be violated by the current action in the protocol (i.e. $\neg viol(x,\alpha,\delta)$ will hold in the next state) if we can show that, at this moment, the obliged action is done and the deadline has not yet passed ($\neg\Diamond\delta$). A proof of this theorem can be found in Appendix A. Note that similar rules could be specified for conditional obligations and (temporal) prohibitions.

Now, let us start the invariance proof by assuming that $\text{start}_\Pi \wedge \Box Norms \rightarrow \neg violation$ holds and try to prove that $\neg violation$ is an invariance of every following step of the protocol, thereby deriving that $\neg violation$ is an invariant of the protocol. Note that we only need to check the actions taken by the protocol, since the 'control points' used in the protocol (i.e. protocol labels referring to conditions of **if**-clauses) are trivially norm-compliant since they do not change the value of any *viol*-predicate (actually, the action that is thereafter chosen shows whether the decision made at the control point was correct). This is expressed in step (1).

(1)	$\neg violation$ invof $\mathscr{M}_{\Pi} \backslash \{\pi_0,\pi_5,\pi_7,\pi_9,\pi_{14},\pi_{16},\pi_{21},\pi_{23},\pi_{24},\pi_{26},\pi_{33}\}$

Next we prove that step π_0 of the protocol (checking whether the patient p satisfies the criteria and none of the contra-indications for being a donor) is norm-compliant. The only norm in the law concerning this action is the fact that doctors are supposed to check whether a patient is brain dead before removing any organs (see article 14, included in Appendix C), of which the translation is seen in step (3). Now, since we can derive from the structure of the protocol that $DO_{d'}\, remove\_organ(d', p)$ has not yet occurred, or is occurring now (5), we can derive that the value of V_1 will not be changed by $DO_d\, certify\_death(d, p)$ by applying the theorem introduced above[4], shown in (6). Finally, after remembering the fact that obligations imply permissions (7) (and therefore do not lead to violations by acting upon the obligation)[5], and adding the fact that no other norms were applicable and thereby cannot be violated (8)[6], we can conclude that $\neg violation$ is an invariant of π_0, see (10).

(2)	at $\pi_0 \rightsquigarrow DO_d\, certify\_death(d,p)$	
(3)	$O_d(DO_d\, certify\_death(d,p) < DO_{d'}\, remove\_organ(d',p))$	Art. 14
(4)	$V_1 = viol(d, DO_d\, certify\_death(d,p), DO_{d'}\, remove\_organ(d',p))$	(Def)
(5)	at $\pi_0 \to \neg \Diamond DO_d\, remove\_organ(d',p)$	(Π)
(6)	at $\pi_0 \wedge \neg violation \to \bigcirc \neg V_1$	(DR),(2),(3),(5)
(7)	at $\pi_0 \to P_d(DO_d\, certify\_death(d,p))$	(3)
(8)	at $\pi_0 \wedge \neg violation \to \bigcirc \neg viol(d,\alpha,\delta)$ for all viol-predicates other than V_1[7]	(7),(VC)
(9)	at $\pi_0 \wedge \neg violation \to \bigcirc \neg violation$	(6),(8)
(10)	$\neg violation$ invof π_0	(9)

The next step (π_5) is an instantiation of the $DO_d\, consult(d, register)$ action mentioned in article 10 of the norms (11). The norm, specified in (12) says that only persons authorised by a doctor have the permission to consult the donor

[4]Application of the derivation rule in theorem 6.14 is referred to by the abbreviation (DR).

[5]Note that although we already assumed that our set of norms is consistent, the derivation shown in (7) only holds if we assume an even stronger restriction on the set of norms. The derivation shown here only holds if the set of norms does not contain any norm conflicts; i.e., the derivation shown does not hold if the set of norms include, next to article 14, a norm as, e.g., $O_d((NOT\ DO_d\, certify\_death(d, p)) < DO_{d'}\, remove\_organ(d', p))$. However, since article 14 is the only norm concerning the *certify_death* action, the derivation shown in (7) is correct.

[6]This process of using the fact that no other norms are applicable and therefore cannot be violated by the protocol step will be used several times in the proof. We call this step 'violation completion' and use the abbreviation (VC) to denote the use of this step.

[7]Note that $viol(d, \alpha, \delta)$ is in fact a disjunction of all violation-predicates other than the violation raised by article 14, i.e., $viol(d, DO_d\, certify\_death(d, p), DO_{d'}\, remove\_organ(d', p)$. We will use this formulation instead of the full disjunction for space saving reasons and to make the expression more readable.

register for information on patients, and then only when this is necessary for the removal of organ. Since we introduced the $\sim P_x(DO_x\,\alpha) \rightarrow F_x(DO_x\,\alpha)$ rule in subsection 6.2.1, we need to check if the conditions of this permission are met, since, if the conditions do not apply, the above mentioned rule specifies that it is actually forbidden to consult the register. This corresponds to our intuitions of the norm; people who are not under the authority of a doctor or when it is not necessary for removing organs, consulting the register is not permitted.Next we use the domain knowledge introduced in subsection 6.4.1 about what it means to for something to be necessary, shown in (14)[8], i.e. not consulting the register would mean that no organs will ever be removed, which is the case according to the specification of the protocol (the protocol would halt and never terminate, expressed in (15)). Using this we can derive the second condition of article 10, shown in (16). The third condition, i.e. being under the authority of a doctor when consulting the register, is trivially true since, in our case, the doctor is running the protocol and therefore always under the authority of a doctor (17). Having shown that the norm is applicable by combining the derived conditions (16) and (17) we can now derive that article 10 is actually applicable and therefore no violation occurs because of it (19).

(11)	at $\pi_5 \rightsquigarrow DO_d\, consult(d,register)$	
(12)	$P_d((DO_d\, consult(d,register)) \mid$ $(under\_authority(d,consult(d,register)) \wedge$ $necessary\_for(consult(d,register),intended(organ\_removal)))$	Art. 10.3
(13)	$V_2{=}viol(d,DO_d\, consult(d,register))$	(Def)
(14)	$necessary\_for(consult(d,register),intended(organ\_removal)) \leftrightarrow$ $\Box(\neg DO_d\, consult(d,register) \rightarrow \neg\Diamond DO_{d'}\, remove\_organ(d',p))$	(DK)
(15)	$\Box(\neg DO_d\, consult(d,register) \rightarrow \neg\Diamond DO_{d'}\, remove\_organ(d',p))$	(Π)
(16)	$necessary\_for(consult(d,register),intended(organ\_removal))$	(14),(15)
(17)	$under\_authority(d,consult(d,register))$	(DK)
(18)	at $\pi_5 \rightarrow P_d(DO_d\, consult(d,register))$	(12),(16),(17)
(19)	at $\pi_5 \wedge \neg violation \rightarrow \bigcirc\neg V_2$	(11),(18)

This, however, is not the only norm concerning step π_5; there is another law regulating this section of the protocol, namely one that obliges doctors to check for the information in the donor register (see article 20.1 in Appendix C). This norm states that doctors who have just certified the death of the patient, are supposed to check whether a patient left a statement of intent concerning the removal of organs, which corresponds to the steps of the protocol from π_4 up to π_9 (the actual actions are of course π_5 and π_7 or π_9, but the intermediate steps are required to determine which actions need to be performed), as seen in (20). The norm is specified in (21) and translated by means of the definition 6.4 into the deadline of (22). Using the fact that step π_0 was considered as doing the $certify\_death$ action (stated in (2)), we can conclude, by looking at the possible

[8]We use (DK) as an abbreviation to show that 'domain knowledge' has been applied in a proof step.

transitions of the protocol, that the state of π_4 always is three states after π_0 (through the transitions $\pi_0; \pi_1; \pi_2; \pi_4$ or $\pi_0; \pi_1; \pi_3; \pi_4$, see the formal protocol in Appendix C for details), shown in (24). Now assuming that the reaction time is at least 4 or 5, as intuition might expect, we can conclude that the deadline has not yet passed, (25), and by applying theorem 6.14 we obtain (26). By splitting the norm compliance over the individual steps we can conclude that the value of *violation* is not changed in any of these steps ((36) up to (41)). Note that we needed to prove the violation-invariance of doing the consult register action (derived in (11)-(19)) as well as of the statement checking action (derived in (20)-(19)) before we could apply the fact that no other norms are violated as stated in (28) and derive that (31) and (37) hold.

(20)	at $\pi_4;\pi_5;\pi_6;\pi_7;\pi_8;\pi_9 \rightsquigarrow DO_d\, check(d,z)$	
(21)	$O_d(DO_d\, check(d,statement(p)) > \Diamond DONE_d\, certify\_death(d,p))$	Art 20.1
(22)	$\Box(DONE_d\, certify\_death(d,y) \rightarrow$	
	$O_d(DO_d\, check(d,z) \leq (\odot^{r.t.}\, DONE_d\, certify\_death(d,p))))$	(21)
(23)	$V_3 = viol(d, DO_d\, check(d,z), DONE_d\, certify\_death(d,p))$	(Def)
(24)	at $\pi_4 \rightarrow \odot^3\, DONE_d\, certify\_death(d,p)$	(2),(Π)
(25)	at $\pi_4 \rightarrow \neg\Diamond\odot^{r.t.}\, DONE_d\, certify\_death(d,p)$	
	For $r.t.$greater or equal to 4	(24)
(26)	at $\pi_4;\pi_5;\pi_6;\pi_7;\pi_8;\pi_9 \wedge \neg violation \rightarrow \bigcirc \neg V_3$	(DR),(21),(20),(25)
(27)	at $\pi_4;\pi_5;\pi_6;\pi_7;\pi_8;\pi_9 \rightarrow P_d(check(d,statement(p)))$	(22),(24)
(28)	at $\pi_5 \wedge \neg violation \rightarrow \bigcirc \neg viol(d,\alpha,\delta)$	
	for all viol-predicates other than V_2 and V_3	(VC)
(29)	at $\pi_4;\pi_6;\pi_7;\pi_8;\pi_9 \wedge \neg violation \rightarrow \bigcirc \neg viol(d,\alpha,\delta)$	
	for all viol-predicates other than V_3	(VC)
(30)	at $\pi_4 \wedge \neg violation \rightarrow \bigcirc \neg violation$	(26),(29)
(31)	at $\pi_5 \wedge \neg violation \rightarrow \bigcirc \neg violation$	(26),(19),(28)
(32)	at $\pi_6 \wedge \neg violation \rightarrow \bigcirc \neg violation$	(26),(29)
(33)	at $\pi_7 \wedge \neg violation \rightarrow \bigcirc \neg violation$	(26),(29)
(34)	at $\pi_8 \wedge \neg violation \rightarrow \bigcirc \neg violation$	(26),(29)
(35)	at $\pi_9 \wedge \neg violation \rightarrow \bigcirc \neg violation$	(26),(29)
(36)	$\neg violation$ invof π_4	(30)
(37)	$\neg violation$ invof π_5	(31)
(38)	$\neg violation$ invof π_6	(32)
(39)	$\neg violation$ invof π_7	(33)
(40)	$\neg violation$ invof π_8	(34)
(41)	$\neg violation$ invof π_9	(35)

Step π_{14} provides us with a strange case, as talking to relatives to inform them what the decisions of the deceased where concerning organ donation does not seem wrong at all, but since we introduced the rule that everything that is not (explicitly) permitted is actually forbidden, this step might actually lead to a violation. There are no norms in the organ donation laws that explicitly, or indirectly, give doctors the permission to inform the relatives. Even so, since an official protocol, which is actually used in the medical domain, includes this action, one would assume that informing the relatives would not violate any of the norms. Therefore, there seems to be something incorrect to just derive that a violation occurs because of this step. Remember, however, that in subsection 6.2.1 we explained, when we discussed why we needed the non-permission rule, that this rule is introduced because the general default of the organ donation domain (being included in the domain of cutting in and operating on people) is not to allow anything unless explicitly permitted. Note then that informing relatives is actually not really a part of that domain, and therefore actually not regulated by the laws of that domain. And since the domain of which informing relatives seems part of seems to be more of the 'opposite' default, i.e., anything is allowed if it is not explicitly prohibited, we can safely assume that step π_{14} does not violate any of the concerning norms.

(42)	$\neg violation$ invof π_{14}	*Fact*

The violation-invariance proof of step π_{16} starts in the usual manner; first we introduce the connection between the abstract and concrete levels (43), and then we introduce the norm concerning this step, which, in this case, states that doctors should, after determining that no statement of intent exists, confer with the relatives of the deceased to obtain the permission for extracting the organs (44). Then, after translating the temporal obligation into a deadline by using definition 6.4 to obtain (46), we start deriving the conditions of theorem 6.14. As it follows from the protocol that the doctor did check for the statement in the past (namely in steps $\pi_4 - \pi_9$), combined with translating (46) using definition 6.3 we can obtain (48). Using the fact that the doctor now knows that p is a potential donor, who is not registered and did not leave another statement of intent, and applying the domain knowledge mentioned in section 6.4.1 that knowledge is truthful, we obtain the desired condition that no statement of intent exists ((49)-(53)). The protocol structure also shows us that the completed *check* action is no more than 3 steps in the past, thus expressing that the deadline of (48) has not yet passed ((54)-(56)). After applying theorem 6.14 we obtain that V_4 is not violated, (57), and that π_{16} is not affecting the value of *violation*, (61).

(43)	at $\pi_{16} \leadsto DO_d\ \mathit{confer\_with}(d,x)$	
(44)	$O_d((DO_d\ \mathit{confer\_with}(d,person) > (\Diamond DONE_d\ \mathit{check}(d,z))) \mid$	
	$((\neg \mathit{statement\_of\_intent}(w,p) \wedge \mathit{relative}(person) \wedge w{=}z) \wedge$	
	$\Diamond DONE_d\ \mathit{certify\_death}(d,p)))$	Art 20.2
(45)	$V_4{=}\mathit{viol}(d,DO_d\ \mathit{confer\_with}(d,x))$	(Def)
(46)	$\Box(\Diamond DONE_d\ \mathit{check}(d,z) \rightarrow$	
	$O_d((DO_d\ \mathit{confer\_with}(d,person) < \odot^{r.t.} \Diamond DONE_d\ \mathit{check}(d,z)) \mid$	
	$((\neg \mathit{statement\_of\_intent}(w,p) \wedge w{=}z \wedge \mathit{relative}(person)) \wedge$	
	$\Diamond DONE_d\ \mathit{certify\_death}(d,p))))$	(44)
(47)	at $\pi_{16} \rightarrow \Diamond DONE_d\ \mathit{check}(d,w)$	(Π)
(48)	$\Box(((\neg \mathit{statement\_of\_intent}(w,p) \wedge w{=}z \wedge \mathit{relative}(person)) \wedge$	
	$\Diamond DONE_d\ \mathit{certify\_death}(d,p)) \rightarrow$	
	$(O_d(DO_d\ \mathit{confer\_with}(d,x) < \odot^{r.t.} \Diamond DONE_d\ \mathit{check}(d,z))$ **until**	
	$(DONE_d\ \mathit{confer\_with}(d,x) \vee \odot^{r.t.} \Diamond DONE_d\ \mathit{check}(d,z)))$	(46),(47)
(49)	at $\pi_{16} \rightarrow \mathit{know\_potential\_donor}(d,p) \wedge \mathit{know\_not\_registered}(d,p) \wedge$	
	$\mathit{know\_not\_exist\_other\_statement}(d,z,p))$	(Π)
(50)	$\mathit{know\_potential\_donor}(d,p) \wedge \mathit{know\_not\_registered}(d,p) \wedge$	
	$\mathit{know\_not\_exist\_other\_statement}(d,z,p)) \rightarrow$	
	$(\mathit{potential\_donor}(p) \wedge \neg \mathit{registered}(p) \wedge \neg \mathit{other\_statement}(z,p))$	Prop. of *known*
(51)	$(\mathit{potential\_donor}(p) \wedge \neg \mathit{registered}(p) \wedge \neg \mathit{other\_statement}(z,p)) \rightarrow$	
	$(\neg \mathit{registered}(p) \wedge \neg \mathit{other\_statement}(z,p))$	$(\mathit{Tautology})$
(52)	$\neg \mathit{registered}(p) \wedge \neg \mathit{other\_statement}(z,p) \rightarrow \neg \mathit{statement\_of\_intent}(w,p)$	(DK)
(53)	at $\pi_{16} \rightarrow \neg \mathit{statement\_of\_intent}(w,p)$	(49)−(52)
(54)	at $\pi_{10} \rightarrow \Diamond DONE_d\ \mathit{check}(d,z)$	(Π)
(55)	at $\pi_{16} \rightarrow \odot^3 \Diamond DONE_d\ \mathit{check}(d,z)$	(54),(Π)
(56)	at $\pi_{16} \rightarrow \neg \Diamond \odot^{r.t.} \Diamond DONE_d\ \mathit{check}(d,z)$	
	For $r.t.$ greater or equal to 4	(55)
(57)	at $\pi_{16} \wedge \neg \mathit{violation} \rightarrow \bigcirc \neg V_4$	(DR),(43),(48),
		(53),(56)
(58)	at $\pi_{16} \rightarrow P_d(\mathit{confer\_with}(d,x))$	(48),(53),(54)
(59)	at $\pi_{16} \wedge \neg \mathit{violation} \rightarrow \bigcirc \neg \mathit{viol}(d,\alpha,\delta)$	
	for all viol-predicates other than V_4	(58),(VC)
(60)	at $\pi_{16} \wedge \neg \mathit{violation} \rightarrow \bigcirc \neg \mathit{violation}$	(57),(59)
(61)	$\neg \mathit{violation}$ invof π_{16}	(60)

At π_{20} the protocol splits two ways, since the doctor has different procedures in case of natural and non-natural deaths. First we will prove that the *announce*-action in π_{21}, which is executed when the patient died of natural causes, is norm-compliant. The law dictates that organs that are supposedly becoming available for transplantation should be announced at an organ centre (c in this case), which is formalised in (63). This is supposed to be done by the doctor who certified the death of the patient. Since π_0 was considered as doing *certify_death*, this condition of the norm is satisfied. Using the definitions of section 6.3 we obtain the deadline necessary for applying theorem 6.14, (65). By using the definition of what it means that the organs have become available (introduced in section 6.4.1), we can show that it follows from the protocol that the deadline of (65) has not yet passed, shown in (71). Again applying theorem 6.14 we obtain that article 18.1 was not violated in the case the patient died of natural causes, (72).

(62)	at $\pi_{21} \rightsquigarrow DO_d\, announce(d,organ(p),c)$	
(63)	$O_d((DO_d\, announce(d,organ(p),c) > available(organ(p))) \mid \Diamonddot DONE_d\, certify\_death(d,p))$	Art 18.1
(64)	$\Box(available(organ(p)) \rightarrow O_d((DO_d\, announce(d,organ(p),c) < \odot^{r.t.} available(organ(p))) \mid \Diamonddot DONE_d\, certify\_death(d,p)))$	(63)
(65)	$\Box(available(organ(p) \rightarrow (\Box(\Diamonddot DONE_d\, certify\_death(d,p) \rightarrow O_d(DO_d\, announce(d,y) < \odot^{r.t.} available(organ(p)))$ **until** $DONE_d\, certify\_death(d,p) \vee \odot^{r.t.} available(organ(p))))))$	(64)
(66)	$V_5 = viol(d, DO_d\, announce(d,organ(p),c))$	(Def)
(67)	at $\pi_{21} \rightarrow \Diamonddot DONE_d\, certify\_death(d,p)$	(Π),(2)
(68)	$((know\_statement\_permission(d,p) \vee know\_other\_statement(d,p) \vee know\_relative\_permission(d,p)) \wedge (\neg non\text{-}natural\_death(p) \vee (non\text{-}natural\_death(p) \wedge know\_DA\_permission(d,p)))) \rightarrow \Box available(organ(p))$	(DK)
(69)	at $\pi_{21} \rightarrow ((know\_statement\_permission(d,p) \vee know\_other\_statement(d,p) \vee know\_relative\_permission(d,p)) \wedge \neg non\text{-}natural\_death(p))$	(Π)
(70)	at $\pi_{21} \rightarrow available(organ(p))$	(68),(69)
(71)	at $\pi_{21} \rightarrow \neg \Diamonddot \odot^{r.t.} available(organ(p))$	(Π),(70)
(72)	at $\pi_{21} \wedge \neg violation \rightarrow \bigcirc \neg V_5$	(67),(DR), (62),(63),(71)
(73)	at $\pi_{21} \rightarrow P_d(DO_d\, announce(d,organ(p),c))$	(65),(67),(70)
(74)	at $\pi_{21} \wedge \neg violation \rightarrow \bigcirc \neg viol(d,\alpha.\delta)$ For all viol-predicates other than V_5	(73),(VC)
(75)	at $\pi_{21} \wedge \neg violation \rightarrow \bigcirc \neg violation$	(72),(74)
(76)	$\neg violation$ invof π_{21}	(75)

The other ‘branch’ of the protocol is used when the patient has died of non-natural causes. In such cases organ donation is not necessarily taking place when permission has been granted by the patient himself or by his relatives, because removing the organs may hinder the investigation towards the death of the patient. In such cases the doctor who certified the death of the patient is not allowed to perform the autopsy of the patient. Autopsies are normally performed at the moment of the removal of the organ, and are used to determine the exact causes of death. If, however, the doctor certifying the death of the patient suspects that his patient has not died of natural causes, he is not allowed to perform this autopsy and is obliged to report to the Municipal Autopsist instead (see article 7.3 of the Law on the disposal of the dead included in Appendix C), stated in (78). Following from the structure of the protocol, we can derive, in (81)-(83), that at π_{23} there actually is a suspicion of a non-natural death, therefore activating the obligation for the doctor to report to the autopsist a. Since the doctor *is* reporting to the autopsist (since that is what π_{23} is all about, (77)), we can derive, by application of theorem 6.14 that no violation will occur because of doing π_{23} at this moment, as stated in (85). Again, remembering that obligations imply permissions, we conclude, since article 7.3 of the Law on the disposal of the dead is the only applicable norm, that π_{23} is non-violation invariant, (89).

(77)	at $\pi_{23} \leadsto DO_d\ \mathit{report\_to}(d,a)$	
(78)	$O_d(DO_d\ \mathit{report\_to}(d,a) > \mathit{suspicion}(d,p,\mathit{non\text{-}natural\_death}))$	Art 7.3 LDotD
(79)	$\Box(\mathit{suspicion}(\mathit{non\text{-}natural\_death}) \rightarrow$ $O_d(DO_d\ \mathit{report\_to}(d,a) < \odot^{r.t}\mathit{suspicion}(d,p,\mathit{non\text{-}natural\_death})))$	(78)
(80)	$V_6 = \mathit{viol}(d, DO_d\ \mathit{report\_to}(d,a), \mathit{suspicion}(d,p,\mathit{non\text{-}natural\_death}))$	(Def)
(81)	at $\pi_{23} \rightarrow \mathit{know\_non\text{-}natural\_death}(d,p)$	(Π)
(82)	$\mathit{know\_non\text{-}natural\_death}(d,p) \rightarrow \mathit{suspicion}(d,p,\mathit{non\text{-}natural\_death})$	(DK)
(83)	at $\pi_{23} \rightarrow \mathit{suspicion}(d,p,\mathit{non\text{-}natural\_death})$	(81),(82)
(84)	at $\pi_{23} \rightarrow \neg\Diamond\odot^{r.t.}\ \mathit{suspicion}(d,p,\mathit{non\text{-}natural\_death})$	(83)
(85)	at $\pi_{23} \wedge \neg \mathit{violation} \rightarrow \bigcirc \neg V_6$	(DR),(78),(77), (83),(84)
(86)	at $\pi_{23} \rightarrow P_d(DO_d\ \mathit{report\_to}(d,a))$	(79),(83)
(87)	at $\pi_{23} \wedge \neg \mathit{violation} \rightarrow \bigcirc \neg \mathit{viol}(d,\alpha,\delta)$ For all viol-predicates other than V_6	(86),(VC)
(88)	at $\pi_{23} \wedge \neg \mathit{violation} \rightarrow \bigcirc \neg \mathit{violation}$	(85),(87)
(89)	$\neg \mathit{violation}$ invof π_{23}	(88)

Like earlier in step π_{14} there is no explicit permission or a direct obligation applicable to the next step in the protocol (π_{24}). Similar as before, the action in π_{24} is not really from the domain of organ donations, and, evenmore, is not actually a part of the model protocol as depicted in figures 6.1 and 6.2. The protocol in these figures merely states that, in the case of a non-natural death, if the district attorney does not give the permission for organ transplantation the organs may not be extracted, and if the district attorney does give permission, extraction can take place (if all other condition are met, of course). This split in the protocol is justified by article 17 of the law (which is included in section Appendix C), which states that it is prohibited for doctors to extract a patient's organs if no permission has been given by the district attorney in the case of a non-natural death. However, from a practical point of view the doctor needs to know what the judgement of the district attorney is, and therefore needs to ask him about it, explaining the explicit formalisation of step π_{24} in the protocol. Since not checking whether the district attorney granted permission for the extraction will ensure that no organs will ever be obtained from this patient, and since it is the organs that the doctor is interested in (his goal is to legally obtain as many organs for transplantation as possible) this step is actually needed (or even obliged) from the hospital's point of view. Therefore, asking a district attorney for whether this particular permission exists, at this moment, does not seem to violate any of the norms, so we can, again, safely assume that π_{24} in non-violation invariant.

(90) $\neg violation$ invof π_{24}	*Assumption*

Earlier we proved that, in the case of a natural death, the obligation to announce available organs to an organ centre was fulfilled at the right moment, see (62)-(76). However, if the patient has died of non-natural causes, the organs have not yet been announced. This is, as we will show, no problem, since the availability of the organs had not yet been established. Similar to the proof of π_{21} we can derive that the doctor has certified the death of the patient earlier, (96). Now, since the permission obtained from the district attorney determines whether the organs will be available for donation (remember step π_{23}-π_{26} are only performed in case of a non-natural death), we can derive (by using the domain knowledge introduced in section 6.4.1 which determines what it means that organs are available, stated in (97)) that the organs have become available for transplantation just now, seen in (97)-(99). Thus, the deadline of the norm in (94) has not passed yet in this case either, and we can, similar to the proof of π_{21} derive, by again using theorem 6.14, that the value of *violation* is not changed by π_{26}.

(91)	at $\pi_{26} \rightsquigarrow DO_d\, announce(d,organ(p),c)$	
(92)	$O_d((DO_d\, announce(d,organ(p),c) > available(organ(p)))\ \mid$ $\Diamond DONE_d\, certify\_death(d,p))$	Art 18.1
(93)	$\Box(available(organ(p)) \rightarrow$ $O_d((DO_d\, announce(d,organ(p),c) < \odot^{r.t.} available(organ(p)))\ \mid$ $\Diamond DONE_d\, certify\_death(d,p)))$	(92)
(94)	$\Box(available(organ(p) \rightarrow (\Box(\Diamond DONE_d\, certify\_death(d,p) \rightarrow$ $O_d(DO_d\, announce(d,p) < \odot^{r.t.} available(organ(p)))$ **until** $DONE_d\, certify\_death(d,p) \vee \odot^{r.t.} available(organ(p))))))$	(93)
(95)	$V_5 = viol(d, DO_d\, announce(d,organ(p),c))$	(Def)
(96)	at $\pi_{26} \rightarrow \Diamond DONE_d\, certify\_death(d,p)$	(Π),(2)
(97)	$((know\_statement\_permission(d,p) \vee know\_other\_statement(d,p) \vee$ $know\_relative\_permission(d,p)) \wedge (\neg non\text{-}natural\_death(p) \vee$ $(non\text{-}natural\_death(p) \wedge know\_DA\_permission(d,p)))) \rightarrow$ $\Box available(organ(p))$	(DK)
(98)	at $\pi_{26} \rightarrow ((know\_statement\_permission(d,p) \vee$ $know\_other\_statement(d,p) \vee know\_relative\_permission(d,p)) \wedge$ $non\text{-}natural\_death(d,p) \wedge know\_DA\_permission(d,p))$	(Π)
(99)	at $\pi_{26} \rightarrow available(organ(p))$	(97),(98)
(100)	at $\pi_{26} \rightarrow \neg \Diamond \odot^{r.t.} available(organ(p))$	(Π),(99)
(101)	at $\pi_{26} \wedge \neg violation \rightarrow \bigcirc \neg V_7$	(96),(DR),(92), (100),(91)
(102)	at $\pi_{26} \rightarrow P_d(DO_d\, announce(d,organ(p),c))$	(94),(96),(99)
(103)	at $\pi_{26} \wedge \neg violation \rightarrow \bigcirc \neg viol(d,\alpha.\delta)$	
	For all viol-predicates other than V_7	(102),(VC)
(104)	at $\pi_{26} \wedge \neg violation \rightarrow \bigcirc \neg violation$	(101),(103)
(105)	$\neg violation$ invof π_{26}	(104)

The final step of the protocol (π_{33}), filling the donor form, is again an action that is not regulated by the norms on organ donation. There are no norms in the law that oblige doctors to fill the donor form, however, it is clear that this action is required for a good functioning of the organisation. So, although filling donor forms is not explicitly permitted or obliged in the Law on Organ Donations, we can give a explanation similar to the justification of steps π_{14} and π_{24}, thus, π_{33} being an action *outside* the domain regulated by the Law on Organ Donations making the prohibition-as-failure rule ($\sim P\alpha \rightarrow F\alpha$) non-applicable. Moreover, filling donor forms after the process described in the protocol can actually be an organisational norm (which are not considered here), which is done in order to keep some record of steps taken and, if necessary, for proving that nothing was done illegally.

(106)	$\neg violation$ invof π_{33}	*Assumption*

Now, after proving that all actions in the protocol are non-violation invariant (when performed at their specified time), we can derive by applying the invariance rule mentioned in theorem 6.13 that the protocol is non-violation invariant. This of course means that the protocol is norm-compliant, since no norms where violated by any of the steps from the protocol, or since no violations occurred during a run of the protocol.

(107)	$\neg violation$ invof $\bar{\mathscr{M}}_\Pi$	(1),(10),(36),(41),(42), (61),(76),(89),(90), (105),(106)

6.5 Proving Liveness

As mentioned in section 6.1, the safety property is not the only property that a protocol needs to be satisfied. Because law is generally applicable to a single context, one who is not participating in the activities of that context is not regulated by these laws; the laws mean nothing to someone not trying to do anything regulated by that particular set of laws. The problem is that laws regulating a specific domain assume that you are trying to do something or otherwise participate in that domain, and only regulates these actions and participations.

While all protocols that satisfy the aforementioned safety property are in fact norm-compliant, it would be nice to show that this protocol is actually goal-oriented as well. To this end, we need to define another property that allows us to determine whether a protocol is, next to being norm-compliant, also trying to achieve something relevant. Norm-compliant protocols that are actually relevant to the domain not only satisfy the violation invariance, but also a liveness property. This sort of properties specifies that a protocol will, at some point, reach a certain (interesting) state. We can use such a property to check whether the protocol achieves a predetermined goal at the end of its run:

Definition 6.15 (Liveness Property of Protocols)

$$\text{start}_\Pi \wedge \Box Norms \quad \rightarrow \quad \Diamond(\text{at } \alpha_e \wedge goal)$$

Where at α_e is the stop-statement of Π and *goal* is the goal that the protocol should reach. In our example this is a complex declarative statement specifying that when the conditions hold (i.e., when the permission for donation is ideally obtained), the agent/doctor running the protocol will know that the donation can take place, and when one of the conditions for the donation fails, the agent/doctor knows that the donation cannot take place.

$$\begin{aligned}
\text{at}\,\alpha_e \;&\equiv\; \text{at}\,\pi_e \\
\mathit{goal} \;&\equiv\; (\mathit{criteria}(p) \wedge \neg \mathit{contra\text{-}indication}(p) \wedge \\
&\qquad (\mathit{registered}(p) \rightarrow \mathit{statement\_permission}(p)) \wedge \\
&\qquad (\neg \mathit{registered}(p) \rightarrow \mathit{other\_statement}(z,p) \vee \mathit{relative\_permission}(p)) \wedge \\
&\qquad (\neg \mathit{non\text{-}natural\_death}(p) \vee \mathit{DA\_permission}(DA,p)) \\
&\qquad \rightarrow \mathit{know\_permission}(d, \mathit{remove\_organ}(p)) \\
&\bigwedge\; \neg(\mathit{criteria}(p) \wedge \neg \mathit{contra\text{-}indication}(p) \wedge \\
&\qquad (\mathit{registered}(p) \rightarrow \mathit{statement\_permission}(p)) \wedge \\
&\qquad (\neg \mathit{registered}(p) \rightarrow \mathit{other\_statement}(z,p) \vee \mathit{relative\_permission}(p)) \wedge \\
&\qquad (\neg \mathit{non\text{-}natural\_death}(p) \vee \mathit{DA\_permission}(DA,p))) \\
&\qquad \rightarrow \mathit{know\_no\_permission}(d, \mathit{remove\_organ}(p))
\end{aligned}$$

This *goal* represents that the protocol is supposed to make sure that the agent obtains the knowledge whether it has the permission for the organ transplantation or not, after ending the protocol run. This permission only exists when the conditions specified are met; i.e. when the patient satisfies the criteria and none of the contra-indications for being a donor ($\mathit{criteria}(p) \wedge \neg \mathit{contra\text{-}indication}(p)$), permission for extraction has been obtained from either the register ($\mathit{statement\_permission}(p)$), another statement of intent ($\mathit{other\_statement}(p)$), or from speaking to the relatives ($\mathit{relative\_permission}(p)$), and permission needs to be obtained from the district attorney as well ($\mathit{DA\_permission}(DA,p)$), if the cause of death was non-natural ($\mathit{non\text{-}natural\_death}(p)$). When these conditions are met the doctor should know, after completing the protocol, that he has the permission to extract the organs of the patient ($\mathit{know\_permission}(d, \mathit{remove\_organ}(p))$). If, however, one of these conditions has not been met, the doctor should know at the end of the protocol that he does not have the permission to remove the organs ($\mathit{know\_not\_permission}(d, \mathit{remove\_organ}(p))$).

This means that if the protocol is run in a situation where the conditions are met, the result should be, at the end of the protocol, that the doctor knows that he has the permission for the extraction. Likewise, if the protocol is run in a situation where one of the conditions has not been met, the result should be that the doctor knows that he does not have permission. This idea leads us to the following theorem that we will use to derive the liveness of the protocol from Appendix C.

Theorem 6.16 (Liveness Rule) *The following rule is valid when γ and $\neg\gamma$ are invariants of* Π.

$$\frac{\begin{array}{c}\text{start}_\Pi \wedge \gamma \rightarrow \Diamond(\text{at}\,\alpha_e \wedge \varphi_1) \\ \text{start}_\Pi \wedge \neg\gamma \rightarrow \Diamond(\text{at}\,\alpha_e \wedge \varphi_2)\end{array}}{\text{start}_\Pi \rightarrow \Diamond(\text{at}\,\alpha_e \wedge (\gamma \rightarrow \varphi_1) \wedge (\neg\gamma \rightarrow \varphi_2))}$$

Theorem 6.16 shows us exactly what we discussed above, namely that if starting the protocol (start_Π) in the situation where the conditions are met (γ) makes sure that at the end of the protocol (π_e) the doctor knows that he has the permission ($know\_permission(d, remove\_organs(p))$), and if starting the protocol in the situation where the condition is not met makes sure that at the end of the run the doctor knows that he does not have the permission, we can conclude that the protocol will, in all situations, give the correct output at the end. A proof of this theorem is included in Appendix A.

In order to prove the assumptions of the rule specified in theorem 6.16, we use the general idea that eventualities like $A \rightarrow \Diamond B$ can be broken down into finitely many 'smaller steps'. The idea would then be to link these smaller steps together into the step required by means of the following rule.

Theorem 6.17 (Chain Rule) *The following rule is valid.*

$$\frac{\begin{array}{c} A \rightarrow \Diamond B \\ B \rightarrow \Diamond C \end{array}}{A \rightarrow \Diamond C}$$

This rule specifies that expressions of the 'smaller steps', e.g. $A \rightarrow \Diamond B$ and $B \rightarrow \Diamond C$, can be chained together to form the 'bigger step', in this case $A \rightarrow \Diamond C$. This process can be repeated to obtain the necessary derivations. To derive the 'smallest steps' required for the application of this rule, we use the following general rule:

Theorem 6.18 (Sometime Rule) *The following rule is valid.*

$$\frac{A \rightarrow \bigcirc B}{A \rightarrow \Diamond B}$$

This rule expresses that if something will happen next, it will happen sometime, which is a general truth following almost directly from the definition of the $\bigcirc$ and $\Diamond$ operators.

6.5.1 Liveness Proof

As mentioned above we are going to make the proof of the liveness property in two parts, first proving that $\text{start}_\Pi \wedge \gamma \rightarrow \Diamond(\text{at}\,\pi_e \wedge \varphi_1)$ holds, with $\varphi_1 \equiv$ $know\_permission(d, y)$ and γ being the conjunction of the following three formulas expressing part of the conditions that should hold:

- $\gamma_1 \equiv criteria(p) \wedge contra\text{-}indication(p)$;
- $\gamma_2 \equiv (registered(p) \rightarrow statement\_permission(p)) \wedge (\neg registered(p) \rightarrow other\_statement(p) \vee relative\_permission(p))$;

- $\gamma_3 \equiv \neg non\text{-}natural\_death(p) \vee DA\_permission(DA, p)$.

The first part is concerning the steps to determine whether the patient satisfies the required criteria and none of the contra-indications of being a donor. Assuming that our model satisfies γ, and thereby satisfies γ_1, the result of the action at π_0 would be that the doctor knows that all the criteria are satisfied and that the patient shows none of the contra-indications (2). The protocol then specifies that the doctor knows at this point that the patient is a potential candidate for extracting organs for transplantation, which is specified in the sequence $\pi_1; \pi_2$ in the protocol, and derived in (3)-(4). By applying the Sometime (theorem 6.18) and Chain rules (theorem 6.17) on steps (2),(3) and (4) we can derive that, if γ holds, we will eventually reach π_4 while knowing that the patient is a potential donor (5).

(1)	$\text{start}_\Pi \wedge \gamma \rightarrow \text{at } \pi_0$	(Π)
(2)	$\text{at } \pi_0 \wedge criteria(p) \wedge \neg contra\text{-}indication(p) \rightarrow$	
	$\bigcirc(\text{at } \pi_1 \wedge know\_criteria(d,p) \wedge know\_no\_contra\text{-}indication(d,p))$	$(PC\ \ \pi_0)$
(3)	$\text{at } \pi_1 \wedge know\_criteria(d,p) \wedge know\_no\_contra\text{-}indication(d,p) \rightarrow \bigcirc \text{at } \pi_2$	(Π)
(4)	$\text{at } \pi_2 \rightarrow \bigcirc(\text{at } \pi_4 \wedge know\_potential\_donor(d,p))$	(Π)
(5)	$\text{start}_\Pi \wedge \gamma \rightarrow \Diamond(\text{at } \pi_4 \wedge know\_potential\_donor(d,p))$	$(SR),(CR),(1)-(4)$

The next part of the proof is a bit more tricky because of the nature of γ_2. Because γ_2 is a disjunction, it can be true in various different ways, which should all be checked. This means that we need to check whether the protocol gives the right response if:

- the patient is registered and has given his permission;
- the patient is not registered but made another statement giving the permission; or
- the patient is not registered, did not make another statement of intent, but the family gives the permission.

First we can derive that we can get to π_5 without much trouble (6), which holds for all instances of γ_2.

(6)	$\text{at } \pi_4 \wedge know\_potential\_donor(d,p) \rightarrow \bigcirc \text{at } \pi_5$	(Π)

Now, let us start by proving the situation when the patient is registered and has given his permission. From doing the check on the register to see if the patient is registered, the doctor knows, in this situation, that the patient is registered after completing the action in π_5, see (7). Now the doctor is supposed to check the registration to see if the patient has actually given permission for organ extraction, (8)-(9), and after performing this action, the doctor is aware of this fact. This influences certain checks, such that protocol gets from π_{10} to π_{13}, π_{14}, π_{15} and finally π_{20}, which follows directly from the structure of the protocol, (10)-(13).

(7)	$\text{at}\,\pi_5 \wedge registered(p) \rightarrow \bigcirc(\text{at}\,\pi_6 \wedge know\_registered(d{,}p))$	$(PC\ \pi_5)$
(8)	$\text{at}\,\pi_6 \wedge know\_registered(d{,}p) \rightarrow \bigcirc \text{at}\,\pi_7$	(Π)
(9)	$\text{at}\,\pi_7 \wedge statement\_permission(d{,}p) \rightarrow$	
	$\bigcirc(\text{at}\,\pi_{10} \wedge know\_statement\_permission(d{,}p))$	$(PC\ \pi_7)$
(10)	$\text{at}\,\pi_{10} \wedge know\_potential\_donor(d{,}p) \wedge know\_registered(d{,}p) \wedge$	
	$know\_statement\_permission(d{,}p) \rightarrow \bigcirc \text{at}\,\pi_{13}$	(Π)
(11)	$\text{at}\,\pi_{13} \wedge know\_potential\_donor(d{,}p) \wedge know\_registered(d{,}p) \wedge$	
	$know\_statement\_permission(d{,}p) \rightarrow \bigcirc \text{at}\,\pi_{14}$	(Π)
(12)	$\text{at}\,\pi_{14} \rightarrow \bigcirc \text{at}\,\pi_{15}$	(Π)
(13)	$\text{at}\,\pi_{15} \wedge know\_possible\_donor(d{,}p) \wedge know\_registered(d{,}p) \wedge$	
	$know\_statement\_permission(d{,}p) \rightarrow \bigcirc \text{at}\,\pi_{20}$	(Π)

Next, in case the patient is not registered, but made another statement of intent to express that he gives the permission for donating his organs, the action at π_5 will result in the doctor knowing that the patient has not registered (14). The doctor now checks, as the protocol specifies, whether another statement exists, which in this case does exist and therefore results in the doctor knowing this, (15)-(17). Knowing this again leads the protocol from π_{10} to π_{13}, π_{14}, π_{15} and π_{20}, again easily obtainable from the structure of the protocol, (18)-(21).

(14)	$\text{at}\,\pi_5 \wedge \neg registered(p) \rightarrow \bigcirc(\text{at}\,\pi_6 \wedge know\_not\_registered(d{,}p))$	$(PC\ \pi_5)$
(15)	$\text{at}\,\pi_6 \wedge know\_not\_registered(d{,}p) \rightarrow \bigcirc \text{at}\,\pi_8$	(Π)
(16)	$\text{at}\,\pi_8 \wedge know\_potential\_donor(d{,}p) \wedge know\_not\_registered(d{,}p) \rightarrow \bigcirc \text{at}\,\pi_9$	(Π)
(17)	$\text{at}\,\pi_9 \wedge other\_statement(z{,}p) \rightarrow \bigcirc(\text{at}\,\pi_{10} \wedge know\_other\_statement(d{,}p))$	(Π)
(18)	$\text{at}\,\pi_{10} \wedge know\_potential\_donor(d{,}p) \wedge know\_not\_registered(d{,}p) \rightarrow \bigcirc \text{at}\,\pi_{13}$	(Π)
(19)	$\text{at}\,\pi_{13} \wedge know\_potential\_donor(d{,}p) \wedge know\_not\_registered(d{,}p) \wedge$	
	$know\_other\_statement(d{,}p) \rightarrow \bigcirc \text{at}\,\pi_{14}$	(Π)
(20)	$\text{at}\,\pi_{14} \rightarrow \bigcirc \text{at}\,\pi_{15}$	(Π)
(21)	$\text{at}\,\pi_{15} \wedge know\_possible\_donor(d{,}p) \wedge know\_not\_registered(d{,}p) \wedge$	
	$know\_other\_statement(d{,}p) \rightarrow \bigcirc \text{at}\,\pi_{20}$	(Π)

In the case, however, that the patient did not register, and did not make another statement of intent, the doctor has to speak to the relative to obtain the permission for extracting the organs. Of course the doctor will first check whether the patient is registered, as in (14)-(16), and after checking whether another statement exists, see (22), he has to conclude that no such statement has been made. The protocol then specifies that the doctor is supposed to speak with the relatives for obtaining the permission, see (23)-(25). After talking to the relatives, who give the permission (which follows from the assumption that γ_2 holds), the doctor knows that the permission has been obtained from the relatives, (26). Finally, the protocol also specifies that the doctor will eventually reach π_{20}, while knowing that the permission has been obtained. Since γ_2 follows from γ, which we assumed to be true, and since in all situations listed above the doctor will eventually reach π_{20} while knowing that the patient is still a potential donor, we can conclude, again by applying the Sometime and Chain rules, that (28) holds.

(22)	$\mathrm{at}\,\pi_9 \wedge \neg \mathit{other\_statement}(z,p) \rightarrow \bigcirc(\mathrm{at}\,\pi_{10} \wedge \mathit{know\_not\_other\_statement}(d,p))$	(Π)
(23)	$\mathrm{at}\,\pi_{10} \wedge \mathit{know\_potential\_donor}(d,p) \wedge \mathit{know\_not\_registered}(d,p) \rightarrow \bigcirc \mathrm{at}\,\pi_{13}$	(Π)
(24)	$\mathrm{at}\,\pi_{13} \wedge \mathit{know\_potential\_donor}(d,p) \wedge \mathit{know\_not\_registered}(d,p) \wedge$ $\mathit{know\_not\_other\_statement}(d,p) \rightarrow \bigcirc \mathrm{at}\,\pi_{15}$	(Π)
(25)	$\mathrm{at}\,\pi_{15} \wedge \mathit{know\_potential\_donor}(d,p) \wedge \mathit{know\_not\_registered}(d,p) \wedge$ $\mathit{know\_not\_other\_statement}(d,p) \rightarrow \bigcirc \mathrm{at}\,\pi_{16}$	(Π)
(26)	$\mathrm{at}\,\pi_{16} \wedge \mathit{relative\_permission}(x,p) \rightarrow$ $\bigcirc(\mathrm{at}\,\pi_{17} \wedge \mathit{know\_relative\_permission}(d,p))$	$(PC\ \pi_{16})$
(27)	$\mathrm{at}\,\pi_{17} \wedge \mathit{know\_relative\_permission}(d,p) \rightarrow \bigcirc \mathrm{at}\,\pi_{20}$	(Π)
(28)	$\mathrm{start}_\Pi \wedge \gamma \rightarrow \Diamond(\mathrm{at}\,\pi_{20} \wedge \mathit{know\_possible\_donor}(d,p))$	(SR),(CR), (5),(6)−(27)

Now that the doctor has determined whether the patient satisfies the criteria for being a donor, and the doctor knows that he has the permission to extract the organs, he needs, in case of a non-natural death, to ask the district attorney for permission to extract the organs. Of course there can be two possibilities; 1) the patient died of natural causes and therefore the doctor can proceed to reporting the organs that are becoming available to the organ centre, see (29)-(30), or 2) the patient died of non-natural causes and the doctor then reports to the municipal coroner for an autopsy, and asks the district attorney for the permission to continuing with the extraction of the organs, (31)-(34). Since we assumed that γ we know that, if the patient died of non-natural causes, the district attorney will give that permission and the doctor will have to announce the organ to the organ centre, (36). Combining these steps, and applying Sometime and Chain once more, we obtain (37), specifying that, in the case that γ holds, the protocol will eventually reach π_{29} while the doctor still knows that the patient is a potential donor.

(29)	$\mathrm{at}\,\pi_{20} \wedge \mathit{know\_potential\_donor}(d,p) \wedge \neg \mathit{non\text{-}natural\_death}(p) \rightarrow \bigcirc \mathrm{at}\,\pi_{21}$	(Π)
(30)	$\mathrm{at}\,\pi_{21} \rightarrow \bigcirc \mathrm{at}\,\pi_{29}$	(Π)
(31)	$\mathrm{at}\,\pi_{20} \wedge \mathit{know\_potential\_donor}(d,p) \wedge \mathit{non\text{-}natural\_death}(p) \rightarrow \bigcirc \mathrm{at}\,\pi_{22}$	(Π)
(32)	$\mathrm{at}\,\pi_{22} \wedge \mathit{know\_potential\_donor}(d,p) \wedge \mathit{non\text{-}natural\_death}(p) \rightarrow \bigcirc \mathrm{at}\,\pi_{23}$	(Π)
(33)	$\mathrm{at}\,\pi_{23} \rightarrow \bigcirc \mathrm{at}\,\pi_{24}$	(Π)
(34)	$\mathrm{at}\,\pi_{24} \wedge \mathit{DA\_permission}(DA,p) \rightarrow \bigcirc(\mathrm{at}\,\pi_{25} \wedge \mathit{know\_DA\_permission}(d,p))$	$(PC\ \pi_{24})$
(35)	$\mathrm{at}\,\pi_{25} \wedge \mathit{know\_DA\_permission}(d,p) \rightarrow \bigcirc \mathrm{at}\,\pi_{26}$	(Π)
(36)	$\mathrm{at}\,\pi_{26} \rightarrow \bigcirc \mathrm{at}\,\pi_{29}$	(Π)
(37)	$\mathrm{start}_\Pi \wedge \gamma \rightarrow \Diamond(\mathrm{at}\,\pi_{29} \wedge \mathit{know\_potential\_donor}(d,y))$	(SR),(CR), (28),(29)−(36)

In steps π_{29} to π_{30}, the protocol specifies that the doctor now knows that he has the permission for extracting the organs. Then in step π_{33} the result of the protocol is logged, and the protocol is stopped in π_{34}. This means that we can derive, using previously derived (37) combined with these small steps, that the protocol will eventually reach the end state when γ holds, and that in such a case the doctor will know that he has the permission to extract the organs, (41).

(38)	$\text{at}\,\pi_{29}\wedge \mathit{know\_potential\_donor}(d,p)\rightarrow\bigcirc\text{at}\,\pi_{30}$	(Π)
(39)	$\text{at}\,\pi_{30}\rightarrow\bigcirc(\text{at}\,\pi_{33}\wedge \mathit{know}_p\mathit{ermission}(d,p))$	$(PC\ \pi_{30})$
(40)	$\text{at}\,\pi_{33}\rightarrow\bigcirc\text{at}\,\pi_{34}$	(Π)
(41)	$\text{start}_\Pi\wedge\gamma\rightarrow\Diamond(\text{at}\,\pi_e\wedge \mathit{know\_permission}(d,p))$	(SR),(CR), (37),(38)−(41)

Having derived the first premise of theorem 6.16, we will now proceed by deriving the second premise of this theorem. To that extend we assume that $\text{start}_\Pi \wedge \neg\gamma$ holds, with $\neg\gamma$ now equivalent to $\neg\gamma_1 \vee \neg\gamma_2 \vee \neg\gamma_3$, and γ_1, γ_2 and γ_3 as specified above. Of course, $\neg\gamma$ can be true in various different ways, which means that the protocol run can be very different from situation to situation and therefore we need to prove for various different cases that the protocol will eventually reach the end state with the desired result.

We start with proving this eventuality for the case that $\neg\gamma_1$. Note that, in this case, it does not really matter whether γ_2 or $\neg\gamma_2$ holds (and likewise for γ_3), since the protocol run is exactly the same for all four combinations (i.e. $\neg\gamma_1 \wedge \gamma_2 \wedge \gamma_3$, $\neg\gamma_1 \wedge \neg\gamma_2 \wedge \gamma_3$, etc). Steps (42)-(46) show what happens in performing π_0 in the case that $\neg\gamma_1$. As mentioned $\neg\gamma_1$ holds if either $\neg\mathit{criteria}(p)$ or *contra-indication*(p) holds which means that either the patient does not satisfy the criteria for becoming a donor, or the patient shows contra-indications for being a donor (such as, for instance, being HIV-positive). And, after performing the check in π_0, the doctor knows this, which is derived in (46).

(42)	$\text{start}_\Pi\wedge\neg\gamma\rightarrow\text{at}\,\pi_0$	(Π)
(43)	$\text{at}\,\pi_0\wedge\neg \mathit{criteria}(p)\wedge\neg \mathit{contra\text{-}indication}(p)\rightarrow$ $\bigcirc(\text{at}\,\pi_1\wedge \mathit{know\_not\_criteria}(d,p)\wedge \mathit{know\_no\_contra\text{-}indication}(d,p))$	$(PC\ \pi_0)$
(44)	$\text{at}\,\pi_0\wedge \mathit{criteria}(p)\wedge \mathit{contra\text{-}indication}(p)\rightarrow$ $\bigcirc(\text{at}\,\pi_1\wedge \mathit{know\_criteria}(d,p)\wedge \mathit{know\_contra\text{-}indication}(d,p))$	$(PC\ \pi_0)$
(45)	$\text{at}\,\pi_0\wedge\neg \mathit{criteria}(p)\wedge \mathit{contra\text{-}indication}(p)\rightarrow$ $\bigcirc(\text{at}\,\pi_1\wedge \mathit{know\_not\_criteria}(d,p)\wedge \mathit{know\_contra\text{-}indication}(d,p))$	$(PC\ \pi_0)$
(46)	$\text{start}_\Pi\wedge\neg\gamma_1\rightarrow$ $\Diamond(\text{at}\,\pi_1\wedge\neg(\mathit{know\_criteria}(d,p)\wedge \mathit{know\_no\_contra\text{-}indication}(d,p)))$	(SR),(CR), (42)−(45)

With the knowledge that the patient does not satisfy the criteria, or shows one or more of the contra-indications of becoming a suitable donor, the protocol specifies that the doctor will know that the patient is not a possible donor, see (47) and (48). This fact makes the rest of the protocol run fairly simple, as most checks in the protocol, which direct the doctor in doing a certain action, are failed and thereby the actions skipped, see (49)-(58). Therefore, we can easily derive that, after obtaining the knowledge that the patient is not a potential donor, the protocol reaches π_e at which time $\mathit{know\_no\_permission}(d,p)$ will hold.

(47)	at $\pi_1 \wedge \neg(know_criteria(d,p) \wedge know_no_contra\text{-}indication(d,p)) \rightarrow \bigcirc$at π_3	(Π)
(48)	at $\pi_3 \rightarrow \bigcirc$(at $\pi_4 \wedge know\_not\_potential\_donor(d,p)$)	($PC\ \pi_3$)
(49)	at $\pi_4 \wedge know_not_potential_donor(d,p) \rightarrow \bigcirc$at π_6	(Π)
(50)	at $\pi_6 \wedge know_not_potential_donor(d,p) \rightarrow \bigcirc$at π_8	(Π)
(51)	at $\pi_8 \wedge know_not_potential_donor(d,p) \rightarrow \bigcirc$at π_{10}	(Π)
(52)	at $\pi_{10} \wedge know_not_potential_donor(d,p) \rightarrow \bigcirc$at π_{13}	(Π)
(53)	at $\pi_{13} \wedge know_not_potential_donor(d,p) \rightarrow \bigcirc$at π_{15}	(Π)
(54)	at $\pi_{15} \wedge know_not_potential_donor(d,p) \rightarrow \bigcirc$at π_{20}	(Π)
(55)	at $\pi_{20} \wedge know_not_potential_donor(d,p) \rightarrow \bigcirc$at π_{22}	(Π)
(56)	at $\pi_{22} \wedge know_not_potential_donor(d,p) \rightarrow \bigcirc$at π_{29}	(Π)
(57)	at $\pi_{29} \wedge know_not_potential_donor(d,p) \rightarrow \bigcirc$at π_{31}	(Π)
(58)	at $\pi_{31} \wedge know_not_potential_donor(d,p) \rightarrow \bigcirc$at π_{32}	(Π)
(59)	at $\pi_{32} \rightarrow \bigcirc$(at $\pi_{33} \wedge know\_not\_permission(d,p)$)	($PC\ \pi_{32}$)
(60)	at $\pi_{33} \rightarrow \bigcirc$at π_{34}	(Π)

Proven that the protocol will, in the case that $\neg\gamma_1$ holds, eventually reach π_e, we have only shown a part of the proof for the case that $\neg\gamma$ holds. Since $\neg\gamma$ can hold when $\neg\gamma_1$ does not we need to explore the other possible valuations as well. In the following steps we assume that $\neg\gamma_1$ did not hold, mean γ_1 holds, and will show that, if $\neg\gamma_2$ holds, the protocol will still reach π_e (with the right conclusion concerning the permission for extracting organs). Since γ_1 holds, the first part of this proof is actually already given in (1)-(6), mentioned earlier.

Remember that

$$\begin{aligned}\neg\gamma_2 \quad &\equiv \quad \neg((registered(p) \rightarrow statement\_permission(p)) \wedge \\ &\qquad (\neg registered(p) \rightarrow other\_statement(p) \vee relative\_permission) \\ &\equiv \quad registered(p) \wedge \neg statement\_permission(p) \vee \\ &\qquad \neg registered(p) \wedge \neg other\_statement(p) \wedge \neg relative\_permission(p)\end{aligned}$$

Like in steps (7)-(28), we split these into separate proofs, showing that whenever $registered(p) \wedge \neg statement\_permission(p)$ holds the eventuality can be proven, as well as whenever $\neg registered(p) \wedge \neg other\_statement(p) \wedge \neg relative\_permission(p)$ holds. We start with the former.

The action performed at π_5 (checking whether the patient is registered) is, in this case, successful, and leads to the doctor knowing that the patient is registered, see (61). The protocol now specifies that the doctor needs to check the registration to see if that registration provides him with the permission needed for extracting the organs, (62) and (63). Since, as mentioned above, $registrated(p) \wedge \neg statement\_permission(p)$ holds, the action at π_7 makes sure that the doctor obtains the knowledge that the patient did not want his organs to be used for transplantation, and therefore, the patient is no longer considered a potential donor, (65) and (66). The rest of the run is as derived in (54)-(60). This means, of course, that the protocol will, in the case of a registered patient who denied the organ extraction, eventually reach π_e at which time the doctor knows that he does not have the permission for extracting the organs.

(61)	$\text{at}\,\pi_5 \wedge \mathit{registered}(p) \rightarrow \bigcirc(\text{at}\,\pi_6 \wedge \mathit{know\_registered}(d,p))$	$(PC\ \pi_5)$
(62)	$\text{at}\,\pi_6 \wedge \mathit{know\_potential\_donor}(d,p) \wedge \mathit{know\_registered}(d,p) \rightarrow \bigcirc\text{at}\,\pi_7$	(Π)
(63)	$\text{at}\,\pi_7 \wedge \neg \mathit{statement\_permission}(p) \rightarrow$ $\bigcirc(\text{at}\,\pi_{10} \wedge \mathit{know\_not\_statement\_permission}(d,p))$	$(PC\ \pi_7)$
(64)	$\text{at}\,\pi_{10} \wedge \mathit{know\_potential\_donor}(d,p) \wedge \mathit{know\_registered}(d,p) \wedge$ $\mathit{know\_not\_statement\_permission}(d,p) \rightarrow \bigcirc\text{at}\,\pi_{11}$	(Π)
(65)	$\text{at}\,\pi_{11} \rightarrow \bigcirc(\text{at}\,\pi_{12} \wedge \neg \mathit{know\_potential\_donor}(d,p))$	(Π)
(66)	$\text{at}\,\pi_{12} \rightarrow \bigcirc(\text{at}\,\pi_{15} \wedge \mathit{know\_not\_potential\_donor}(d,p))$	(Π)

In the case that $\mathit{registered}(p) \wedge \neg\mathit{statement\_permission}(p)$ does not hold, $\neg\mathit{registered}(p) \wedge \neg\mathit{other\_statement}(p) \wedge \neg\mathit{relative\_permission}(p)$ should hold, since we assumed that γ_2 holds. In the following steps we will show that in this case the eventuality will hold as well.

In this case, the action performed at π_5 clearly has the result of the doctor knowing that the patient is not registered, (67). The protocol then specifies that the doctor should check whether another statement of intent exists, (68)-(69). Since this statement does not exist, and the doctor will know this after performing the action in π_9, as shown in (70), the protocol specifies that the doctor should talk to the relatives to see if they provide the permission for the extraction, (71)-(73). Now, since we assumed that the relatives do not give the permission, and the doctor will know this after performing π_{16}, the protocol once again specifies that the patient is not a potential donor. Knowing this, the run continues like before, specified in steps (55)-(60).

(67)	$\text{at}\,\pi_5 \wedge \neg\mathit{registered}(p) \rightarrow \bigcirc(\text{at}\,\pi_6 \wedge \mathit{know\_not\_registered}(d,p))$	$(PC\ \pi_5)$
(68)	$\text{at}\,\pi_6 \wedge \mathit{know\_not\_registered}(d,p) \rightarrow \bigcirc\text{at}\,\pi_8$	(Π)
(69)	$\text{at}\,\pi_8 \wedge \mathit{know\_potential\_donor}(d,p) \wedge \mathit{know\_not\_registered}(d,y) \rightarrow \bigcirc\text{at}\,\pi_9$	(Π)
(70)	$\text{at}\,\pi_9 \wedge \neg\mathit{other\_statement}(p) \rightarrow \bigcirc(\text{at}\,\pi_{10} \wedge \mathit{know\_not\_other\_statement}(d,p))$	$(PC\ \pi_9)$
(71)	$\text{at}\,\pi_{10} \wedge \mathit{know\_not\_registered}(d,p) \rightarrow \bigcirc\text{at}\,\pi_{13}$	(Π)
(72)	$\text{at}\,\pi_{13} \wedge \mathit{know\_not\_registered}(d,p) \wedge \mathit{know\_not\_other\_statement}(d,p) \rightarrow \bigcirc\text{at}\,\pi_{15}$	(Π)
(73)	$\text{at}\,\pi_{15} \wedge \mathit{know\_not\_registered}(d,p) \wedge \mathit{know\_not\_other\_statement}(d,p) \rightarrow \bigcirc\text{at}\,\pi_{16}$	(Π)
(74)	$\text{at}\,\pi_{16} \wedge \neg\mathit{relative\_permission}(d,p) \rightarrow$ $\bigcirc(\text{at}\,\pi_{17} \wedge \mathit{know\_not\_relative\_permission}(d,p))$	$(PC\ \pi_{16})$
(75)	$\text{at}\,\pi_{17} \wedge \mathit{know\_not\_relative\_permission}(d,p) \rightarrow \bigcirc\text{at}\,\pi_{18}$	(Π)
(76)	$\text{at}\,\pi_{18} \rightarrow \bigcirc(\text{at}\,\pi_{19} \wedge \neg\mathit{know\_potential\_donor}(d,p))$	(Π)
(77)	$\text{at}\,\pi_{19} \rightarrow \bigcirc(\text{at}\,\pi_{20} \wedge \mathit{know\_not\_potential\_donor}(d,p))$	(Π)

Having shown that the eventuality holds for the cases where $\neg\gamma_1$ or $\neg\gamma_2$, we only need to show that it also holds if both of $\neg\gamma_1$ and $\neg\gamma_2$ does not hold, i.e. proof the eventuality when $\gamma_1 \wedge \gamma_2$. Since we assumed that $\neg\gamma$, we know that if $\gamma_1 \wedge \gamma_2$ holds, it should hold that $\neg\gamma_3$. In such a case the first steps of the protocol will go as proven above in (1)-(28). Using the fact that $\mathit{non\text{-}natural\_death}(p) \wedge \neg\mathit{DA\_permission}(DA, p)$ combined with the protocol structure, we can easily derive that the run will reach π_{29} sometime, at which time the doctor knows that the patient is not a potential donor, due to the fact that performing π_{24} results in knowing that the district attorney did not give permission to continue with the organ extraction (as was implied by $\neg\gamma_3$). This is shown in the steps (78)-(84). Since the doctor now knows the patient is not a potential donor, the run will end as derived before in (57)-(60).

(78)	$\text{at}\,\pi_{20}\wedge \mathit{know\_potential\_donor}(d,p)\wedge \mathit{know\_non\_natural\_death}(d,p)\rightarrow\bigcirc\text{at}\,\pi_{22}$	(Π)
(79)	$\text{at}\,\pi_{22}\wedge \mathit{know\_potential\_donor}(d,p)\wedge \mathit{know\_non\_natural\_death}(d,p)\rightarrow\bigcirc\text{at}\,\pi_{23}$	(Π)
(80)	$\text{at}\,\pi_{23}\rightarrow\bigcirc\text{at}\,\pi_{24}$	(Π)
(81)	$\text{at}\,\pi_{24}\wedge\neg \mathit{DA\_permission}(DA,p)\rightarrow\bigcirc(\text{at}\,\pi_{25}\wedge \mathit{know\_not\_DA\_permission}(d,p))$	$(PC\ \ \pi_{24})$
(82)	$\text{at}\,\pi_{25}\wedge \mathit{know\_not\_DA\_permission}\rightarrow\bigcirc\text{at}\,\pi_{27}$	(Π)
(83)	$\text{at}\,\pi_{27}\rightarrow\bigcirc(\text{at}\,\pi_{28}\wedge\neg \mathit{know\_potential\_donor}(d,p))$	(Π)
(84)	$\text{at}\,\pi_{28}\rightarrow\bigcirc(\text{at}\,\pi_{29}\wedge \mathit{know\_not\_potential\_donor}(d,p))$	(Π)

Combining the cases when $\neg\gamma_1$, $\neg\gamma_2$ and $\neg\gamma_3$ to the general case that $\neg\gamma$, shown in (85), (86), (87) and (88), we can apply the liveness rule as specified in theorem 6.16 to obtain the general liveness property we set out to derive, shown in (89).

(85)	$\text{start}_\Pi\wedge\neg\gamma_1\rightarrow\Diamond(\text{at}\,\pi_e\wedge \mathit{know\_not\_permission}(d,\mathit{remove\_organ}))$	$(SR),(CR),(42)-(60)$
(86)	$\text{start}_\Pi\wedge\neg\gamma_2\rightarrow\Diamond(\text{at}\,\pi_e\wedge \mathit{know\_not\_permission}(d,\mathit{remove\_organ}))$	$(SR),(CR),(1)-(6),$ $(61)-(66),(67)-(77),$ $(55)-(60)$
(87)	$\text{start}_\Pi\wedge\neg\gamma_3\rightarrow\Diamond(\text{at}\,\pi_e\wedge \mathit{know\_not\_permission}(d,\mathit{remove\_organ}))$	$(SR),(CR),(1)-(28),$ $(78)-(84),(57)-(60)$
(88)	$\text{start}_\Pi\wedge\neg\gamma\rightarrow\Diamond(\text{at}\,\pi_e\wedge \mathit{know\_not\_permission}(d,\mathit{remove\_organ}))$	$(85),(86),(87)$
(89)	$\text{start}_\Pi\rightarrow\Diamond(\text{at}\,\pi_e\wedge(\gamma\rightarrow \mathit{know\_permission}(d,\mathit{remove\_organ}))\wedge$ $(\neg\gamma\rightarrow \mathit{know\_not\_permission}(d,\mathit{remove\_organ})))$	$(LR),(41),(88)$

6.6 Chapter Conclusions

In highly-regulated domains where protocols provide an invaluable means for the agents to act in a norm-compliant manner, it is very important that the protocols and procedures that the agents use are norm-compliant themselves. To ensure that protocols are norm-compliant, a formal verification of the protocol is required to check if no violations ever occur during the run of a protocol.

Although we use a method based on verification techniques for concurrent sequential programs, the process of verifying norm compliance of protocols is very different from the verification of programs, as the emphasis of our technique is on making clear how the protocol is linked to the norms and what concepts used in the protocol and norms mean. As mentioned in 6.2.3, after the verification process as given in section 6.3, we end up with a list of 'assumptions' under which the protocol can be proven to be norm-compliant or norm-breaking. If these assumptions are accepted, the conclusion of the verification process has to be accepted as well. This is very similar to the legal practice in court, where a verdict is dependent on the interpretation of the situation at hand.

The framework presented in this chapter uses a theorem proving method to verify the norm compliance of protocols. This is known to be labour-intensive, and it might be possible to use model-checking techniques for the verification process instead (for example, as done in [Hommersom et al., 2004]). This does not, however, simplify the representation process, which will still have to be done by hand.

Chapter 7

Conclusion

This dissertation has focussed on the design and implementation of electronic institutions based on human laws and regulations. The design of such highly-regulated institutions required the implementation of norm enforcement from an institutional point of view, and the design of protocols that can be used to guide the agents participating in the institution. The motivation for this research came from the ANITA project that aimed to implement a multiagent system in the highly-regulated domain of police registers to solve issues that exist in information exchange between different police districts. Unfortunately, a change in laws and regulations, during the research project, already solved these problems, and we had to switch to a different but more suitable domain.

The formalism presented in this thesis creates the links between norms extracted from human laws and regulations and agent-mediated electronic institutions. Although we mainly focussed on highly-regulated environments (such as the original ANITA scenario), where complex and abstract norms govern all the interactions in the society, the formalism can also be used for the implementation of normative institutions that consist of fewer norms, though, in that case, more information from the practice needs to be added.

This final chapter discusses the results of this project, in section 7.1, and points of future research in section 7.2.

7.1 Discussion of Results

Designing and implementing electronic institutions from law is a difficult process. While research has been done on how to express and reason about norms and how to implement electronic institutions, not much work has been done on the connection of these two; how to create an institution from law. To create a connection between the specifications in laws and regulations on the one hand, and the electronic institutions on the other, we had to solve the following problems.

1. Abstractness of human regulations.
2. Operationalisation of norms.
3. Implementation of norms from the institutional perspective.
4. Creating a methodology to design electronic institutions.

Norms expressed in laws and regulations are stated in an abstract manner to allow the norm to cover a lot of different situations, reducing the need for modifications over long periods of time. This abstractness, however, poses a problem when trying to implement the norm in multiagent systems, where the meaning of the norm should be precise and unambiguous (issue 1). Moreover, norm representations, as spoken of in chapter 2, usually formalise norms from a declarative point of view. These representations express what is right and wrong, usually in terms of obligations, permissions and prohibitions, but lack connections to the operational impact of the norm on the behaviour of the agents in the environment. When implementing norms, however, an operational meaning of the norms is needed (issue 2). This operational meaning defines what has to be done when a norm has been violated or how to check when a norm is violated, thus expressing how the norm should be implemented. Although it would be desirable to have only agents joining the system that can understand and reason about the norms, such an assumption cannot be made in open environments. Instead, to allow agents to join that do not have those abilities, and to be able to handle agents that intentionally try to violate the norms of the system, the compliance to the norms has to be ensured by the institution itself (issue 3). Moreover, since it cannot be guaranteed that agents joining the system are able to understand and reason about the norms, the conventions and rules of the institution have to be conveyed to them in some other (understandable) manner, e.g. by means of the specification of protocols for the different tasks in the institution.

These issues, driven by the motivation to design and implement electronic institutions based on human laws and regulations, resulted in a set of goals for this dissertation.

1. Demonstrate how norms extracted from human laws and regulations are linked to electronic institutions.
2. Develop an implementation of norms from the institutional perspective.
3. Create a method to help the design of protocols for highly-regulated domains.
4. Show how the norm compliance of protocols can be proven formally.

In the remainder of this section, we look back at the realised work, and discuss the main findings related to these objectives.

7.1.1 Relating Law and Institutions

The first goal, which was derived from our first research question presented in chapter 1, deals with the connection between law and electronic institutions. The relation between the norms derived from human laws and regulations and the electronic institution based on these norms, are important for understanding how electronic institutions can be made on the basis of such normative specifications. In chapter 3 we have shown a methodology to build institutions from a specification given by law and regulations. By means of this methodology we have shown that the norms, extracted from human laws and regulations, are important for electronic institutions in different manners. The main element in the design of electronic institution is the (formal) specification of the norms.

The relations between the laws and the electronic institution then become apparent from the process of designing the electronic institution as given by the methodology presented in chapter 3. First, the laws and regulations themselves, and the process of translating them to some formal representation, provides us the 'base' ontology of the institution. This ontology is important for defining the meaning of the concepts used in the institution and in interactions between the participating agents. Second, the norms provide the behaviour restrictions imposed on the agents. The norms express the wanted (legal) and unwanted (illegal) behaviour of the agents, usually expressed in terms of obligations, permissions and prohibitions. These constraints, describing the ideal situation, have to be imposed on the agents by some form of enforcement. Third, to assist agents that do not have norm reasoning capabilities, and to provide a general means to achieve frequently done tasks, protocols need to be created to show the agents how things are normally done in the institution.

While the methodology is not complete, it makes the first steps that are needed and tries to cover the most important aspects of institution design. Unfortunately, there is not much research available on the topic of institution design; most of the important research is done by the people who design legal systems and work on legislation and the creation of laws, but that knowledge is too abstract to be of any use to the concrete solutions. The methodology created in chapter 3 defines a framework for institutions on the basis of the law. This framework consists of an ontology for the institution, partially taken from the law and partially taken from practice; a normative specification derived from the law; and the protocols and norm enforcement, both created on the basis of the normative specification. In chapters 4 and 5 we have looked in more detail to the processes of implementing norm enforcement and designing protocols, respectively.

7.1.2 Implementing Norms

In a discussion of the different manners of implementing norms in agent systems, i.e. either by adopting an agent's or institution's perspective and choosing to hard-code, regiment or enforce the norms by means of violations and sanctions, we concluded that the implementation of norms by means of an active norm enforcement, using violation detection and sanctioning, from an institutional point of view is advantageous for several reasons:

1. Due to the nature of open agent systems, where no assumptions can be made on the inner workings of the agents, it is unwarranted to assume that all agents have the capability to reason about the norms (and the intention to comply with them). In such environments it is the institution itself that needs to ensure that the agents comply with the norms.

2. In order to increase the level of autonomy of the agents participating in the institution, thereby increasing the adaptability and possibly the efficiency of the agents, a norm enforcement mechanism is required where the agents are not bound to a predefined set of protocols, unable to choose different courses of actions than those that were thought of at the design of the institution.

To achieve these advantages, a new sort of norm implementation was required, where norms define the checks to be made (to decide whether the norm is violated) and the reactions that should follow violations of the norm (e.g. to punish the violator). Since this sort of information is not included in the representation of the norms, usually done in a deontic formalism, we had to annotate the representation of the norms with additional fields that are required for the implementation. This information added to the norms is the following:

1. Violation Condition

2. Detection Mechanism

3. Sanction

4. Repairs

The violation condition is added to the norms to have a clear representation of the states that violate the norms. As mentioned above, norms are usually very abstract and it can be hard to link them to the states of the institution that violate the norm (issue 1). By contextualising the norm, and representing the violation states in clear and concrete terms, the detection of the violations is done much easier. This also concerns the second field added to the norms. The detection mechanism is a list of the mechanisms present on the agent platform that can be employed in the detection of the violation. In effect, after an enforcing agent has completed the plan listed in the detection mechanism field, the enforcer knows, for sure, whether the violation condition holds, thus being sure about whether the norm is violated or not and whether responses are required. The responses to the norm violations are listed in the sanction and repairs field, with the former defining the punishments applied to the violating agent to direct their (future) behaviour, and the latter being the actions necessary to revert (or rather fix) the negative side-effects of the action(s) that caused the violation of the norm.

The norms expressed in this operational representation can then be easily translated to operational constraints that can be used by the agents responsible for the enforcement of the norms. In section 4.3 we identified two sorts of constraints; 1) integrity constraints, used for 'flagging' the violations of the norms, thus telling the enforcers that a response is warranted; 2) dialogical constraints, that express the responses that the enforcers must take to deal with 'flagged' violations of the norms.

This process of annotating the norms with operational information, and translating them to operational constraints (combined with the proposed implementations of [Aldewereld et al., 2006b; García-Camino and Rodríguez-Aguilar, 2006]) completes the implementation of the norms from an institutional perspective. As argued above, through the use of violation detection and sanctioning, and using institutional agents responsible for the enforcement, we gain the advantage of a minimal loss of autonomy of the participating agents, as was stated in the second objective.

7.1.3 Creating Protocols

Even though the safety and reliability of the system is ensured by the implementation of an active norm enforcement as mentioned in the previous paragraph, a set of protocols is required to make agents who do not have the capability of reasoning about the norms, able to perform in the institution. The active norm enforcement mentioned above only sees to it that such agents are punished for the violations of the norms, making it hard for them to function efficiently. To complete their (socially accepted) goals agents participating in those institutions either need to be able to reason about the norms, and use the norms to derive which (sequences of) action(s) are legal to try to obtain their goal, or guidelines (protocols) need to be specified to help the agents achieve (the most common) tasks.

However, the design of protocols for domains regulated by lots of norms is a very hard process, due to the gap between the abstract nature of the norms and the lack of a clear connection to the practice. To help solve this problem we introduced, based on an idea taken from the structure of laws in human institutions, an intermediate level between the norms and the practice. This intermediate level works in a similar fashion as regulations in Roman-Germanic legal systems. Regulations are used in such systems to add procedural information (information on how certain states mentioned in the law are to be achieved/avoided) to the existing, but abstract, information expressed in the law. The procedure for designing protocols presented in chapter 5 uses this idea, where the intermediate level of landmarks, that already contains all the important information extracted from the norms, is strengthened with additional landmarks to increase the efficiency and feasibility of the normative landmark pattern. In this manner, the normative landmark pattern, an ordering of important states created based solely on the information taken from the norms, is made one step more concrete by the introduction of procedural landmarks. This reduces the abstractness of the information expressed in the norms, making it easier to create a protocol based on this strengthened landmark pattern, which can thus be seen as a ‘skeleton’ or ‘prototypical’ protocol.

Moreover, we have shown in chapter 5 a semi-automatic technique, based on a procedure designed for model-checking, to extract normative landmarks from a specification of the norms. This process translates the norms expressed in LTL to a finite-state automata representing all the models for that LTL-expression, which can be used to extract the characteristics of the norms; “the important states that should be reached”. This collection of states, and the order in which these states have to be achieved, make up the normative landmark pattern, i.e. the restrictions expressed by the norms about which states should appear in every protocol of that domain, and the order in which these states have to be achieved. We have also shown how a landmark pattern extracted from the norms can be enhanced with procedural information, giving a prototypical protocol for the domain. Moreover, we have shown how this, still abstract, protocol can be translated, by using the expected capabilities of the agents in the institution, to a concrete protocol that can be used in the institution, thus achieving our third objective.

7.1.4 Verifying Norm Compliance

The protocols designed by the method explained in chapter 5 have a certain level of norm compliance guaranteed, because the information form the norms was used directly to design the protocol. When protocols are taken from the real-world practice, or adapted protocols from different sources are used instead, this norm compliance cannot be guaranteed. In such cases it is important to check that the protocol does not violate any of the norms of the domain. To make sure that a protocol does not break any of the norms of the institution at any time, the norm compliance of the protocol must be (formally) verified.

In chapter 6 we presented a formal method for the verification of norm compliance of protocols. This formal method was based on a technique used for the verification of properties of concurrent sequential programs. Protocols, which are in essence not very different from programs, can be verified in a similar manner by checking whether the protocol satisfies certain properties. To verify the protocol we use two different properties; a safety property is specified to denote that no changes should occur to the value of violation, i.e. none of the violation propositions, denoting that a violation of a norm has occurred, should be (or become) true anywhere during the run of the protocol; a liveness property is specified to denote the goal of the protocol to check whether the protocol actually tries to achieve the task it was created for. The norm compliance of the protocol is proven by checking whether the protocol satisfies both these properties.

Although we use a method based on verification techniques for concurrent sequential programs, the process of verifying norm compliance of protocols is very different from the verification of programs, as the emphasis of our technique is on making clear how the protocol is linked to the norms and what the concepts used in the protocol and norms mean. After the verification process we end up with a list of assumptions and choices under which the protocol can be proven to be norm-compliant or norm-breaking. If these assumptions are accepted, the conclusion of the verification process has to be accepted as well. This is very similar to the legal practice in court, where a verdict is dependent on the interpretation of the situation at hand.

7.2 Future Research

Our ideas for future research concern the link between abstract and concrete concepts in the ontology of the institution, implementation of the different aspects, integration with other frameworks, and the use of model-checking and implementation of a tool for the verification of norm compliance.

7.2.1 Normative Ontologies

The methodology presented in chapter 3 gives our current view on the relation between human laws and institutional ontologies. The ontology of institutions is built from the concepts used in the norms and the concepts that are taken from the practice. It is important that these concepts are linked in order to give an institutional meaning to the abstract concepts used in the law. That is, the concepts taken from the norms do not have a clear meaning in the agent practice, and a translation to more concrete concepts is needed for expressing the intended meaning of the norms in the implementation of the institution.

We already shown in chapter 3 that such connections between abstract and concrete concepts can be made through a counts-as operator, but refrained from giving a full formal meaning of this operator. It is thus needed to formally define the meaning of this operator, thereby finally solving issue 1 as mentioned on page 176.

A fairly recent proposal in this direction is given in [Grossi et al., 2004, 2006b,c,d]. This work investigates all formal aspects of the counts-as relation between abstract and concrete terms, linking abstract norms to concrete situations. We plan in the future to investigate the integration of this proposal in our methodology to solidify the translation of norms to an institutional ontology.

7.2.2 Implementing Normative Systems

Most of the work presented in this thesis is of a theoretic nature, although some implementations were proposed here and there. The main subject of this thesis, the method of designing and implementing electronic institutions based on (human) laws, has not (yet) been subjected to a full implementation to create the tools to ease the process of the design of normative systems.

We did propose an extension to AMELI, the current implementation of ISLANDER institutions, to make it able to enforce norms by means of violation detection and sanctioning, as proposed in chapter 4. As we mentioned already, this is, however, not necessarily the only good implementation of the norm frame we presented in that chapter, but it shows how the norm frame can be used to create the mechanisms for institutions to actively enforce the norms. More complex implementations, using all aspects of the norm frame, will have to be researched and created.

Another aspect of this thesis that is interesting for implementation is the creation of a tool to help institution designers with the creation of protocols based on the specification of the norms. We have shown in chapter 5 that the process from a normative specification (in LTL) to protocols can be, at least partially, automated, making this technique very useful in the process for designing protocols (e.g. for designing interaction structures as used in ISLANDER) when implementing an institution.

And lastly, though not done because of changes to the domain, we would have liked to use the theory presented in this thesis to design and implement an electronic institution to solve the problems (now solved by the changes in the law) that existed in the ANITA domain. Implementing an electronic institution in such a domain, based on human laws and regulation, for the actual use in practice would have been an interesting endeavour.

7.2.3 Integrating Organisations

As mentioned in chapter 2, the formalism presented in this thesis is an extension of the work presented in [Vázquez-Salceda, 2004], going into more detail about the relations between the different aspects of institutions, and giving a closer look to the process of implementing norm enforcement and protocol design. As we concluded earlier, our formalism is not complete. One of the points that is not addressed in our approach is the organisational point of view to institutions, as we only focus on the norms and their impact on the implementation.

In [Vázquez-Salceda et al., 2004] a framework is proposed that covers both the normative aspects of institutions (based on the HARMONIA framework of [Vázquez-Salceda, 2004]) and the organisational aspects of institutions (based on the OperA framework of [Dignum, 2004]). It would be interesting to extend this framework with the findings of the research presented in this thesis.

7.2.4 Model-Checking Norm Compliance

In chapter 6 we presented a formal technique for the verification of norm compliance of protocols. The framework we used there was based on a verification technique for concurrent sequential programs, and uses a theorem proving method to verify the properties that the program (or in this case, the protocol) needs to satisfy. It is, however, well-known that theorem proving can be very labour-intensive, and very hard to automate.

If the technique presented in chapter 6 was ever to be implemented to make it possible to verify interaction patterns used in an institution in a (semi-)automatic manner, a more efficient manner of proving the norm compliance has to be used. Such a manner of verification might be found in model-checking, which is presumed to be easier to automate and implement than theorem proving. As we mentioned earlier though, a full implementation of the process described in chapter 6 would be impossible, even when using model-checking, due to the many representational changes that have to be made during the verification process, a semi-automated verification process might be possible.

In that case, the verification of the protocol, done by either model-checking or theorem-proving as presented in chapter 6, would be done 'interactively'. This means that the tool would assist the user in making the proof for the norm compliance, requiring only the input of the representational choices and counts-as links as given in the 'assumptions' list as used in chapter 6 (either given at forehand or during the run of the tool). This, however, still requires some further research before an actual implementation as envisioned here can be made.

7.3 Concluding Thoughts

At the start of this thesis we set out to create a formalism for the design and implementation of agent-mediated institutions based on norms taken from human laws and regulations. While creating this formalisation we looked at the relations between laws and electronic institutions, and designed methods for the design and verification of protocols. Moreover, we investigated the implementation of norms to create a mechanism of active norm enforcement to ensure a good balance between the autonomy and the conformity of the agents participating in the system.

Appendix A

Selected Proofs

In this appendix we proof some of the propositions and theorems used throughout the thesis.

Proofs of LTL-reductions of chapter 4

In chapter 4.2.1 we introduced a linear-time temporal logic for the expression of norms. There we have given semantical definitions of various normatives that are expressable in the logic mentioned. We claimed that these normatives could be reduced to expressions containing only temporal operators already in the formalism, and we will prove some of these reductions here.

Reduction of Deadlines

In definition 4.5 we defined deadlines as the following:

$$\begin{array}{l}\mathcal{M}, s \models \mathsf{OBLIGED}(a, P \; \mathsf{BEFORE} \; D) \quad \Leftrightarrow_{def} \\ \qquad \exists t \geq s : \mathcal{M}, t \models D \text{ and} \\ \qquad (\forall s \leq u < t : \mathcal{M}, u \models \mathsf{NOT}\, viol(a, P, D)) \text{ and} \\ \qquad ((\exists s \leq v < t : \mathcal{M}, v \models P \text{ and } \mathcal{M}, v \models \mathsf{ALWAYS}\,\mathsf{NOT}\, viol(a, P, D)) \text{ or} \\ \qquad (\forall s \leq w < t : \mathcal{M}, w \not\models P \text{ and } \mathcal{M}, t \models viol(a, P, D)))\end{array}$$

We now show that this can be reduced to the LTL formula as presented in proposition 4.6. Assuming that $\mathcal{M}$ at state s satisfies $\mathsf{OBLIGED}(a, P \; \mathsf{BEFORE} \; D)$, i.e. the LTL model complies with the semantical definition given above, we will show that these models satisfy the proposition 4.6 as well.

The first condition of definition 4.5, $\exists t \geq s : \mathcal{M}, t \models D$, tells us that D will hold in some world in the future (with s the current world), i.e. $\mathcal{M}, s \models \mathsf{SOMETIME}\, D$. Although D may hold many times more in the future, we are mainly interested

in its first occurrence:
$\mathcal{M}, s \vDash \mathsf{SOMETIME}\, D\, \mathsf{AND}\, (\mathsf{NOT}\, D\, \mathsf{UNTIL}\, D)$

From the second condition, $\forall s \leq u < t : \mathcal{M}, u \vDash \mathsf{NOT}\, viol(a, P, D)$, we know that no violations will occur before the deadline occurs (in time moment t), i.e. $\mathcal{M}, s \vDash \mathsf{SOMETIME}\, D\, \mathsf{AND}\, ((\mathsf{NOT}\, D\, \mathsf{AND}\, \mathsf{NOT}\, viol(a, P, D))\, \mathsf{UNTIL}\, D)$.

The last condition then specifies how the deadline handles the compliance and violation, respectively. This condition defines two different cases:

1. Conditions 1 and 2 hold until the (first) appearance of P (strictly before the appearance of D), after which no violation of the deadline can occur; or
2. Conditions 1 and 2 hold until the deadline holds and a violation occurs.

Let us first look at the case defining the violation first. $\forall s \leq w < t : \mathcal{M}, w \nvDash P$ and $\mathcal{M}, t \vDash viol(a, P, D)$ tells us that, before t has happened (the time-moment of the deadline), P will never occur, and, in that case, t satisfies $viol(a, P, D)$. Combined with our currently derived formula (for conditions 1 and 2) we get:

$$\begin{array}{l}\mathcal{M}, s \vDash \mathsf{SOMETIME}\, D\, \mathsf{AND} \\ \quad \big[(\mathsf{NOT}\, D\, \mathsf{AND}\, \mathsf{NOT}\, P\, \mathsf{AND}\, \mathsf{NOT}\, viol(a, P, D))\, \mathsf{UNTIL} \\ \quad (D\, \mathsf{AND}\, viol(a, P, D))\big]\end{array}$$

However, if the deadline is adhered to, which is expressed in the first part of the last condition, there is a state v before t (but after s) where P holds, and after that $viol(a, P, D)$ never holds (note that, because v happens strictly before t, v also satisfies $\mathsf{NOT}\, D$, as expressed in the second condition of definition 4.5). This leads to the following when combined with the formula for conditions 1 and 2 (assuming that the occurrence of P is the first occurrence[1]):

$$\begin{array}{l}\mathcal{M}, s \vDash \mathsf{SOMETIME}\, D\, \mathsf{AND} \\ \quad \big[(\mathsf{NOT}\, D\, \mathsf{AND}\, \mathsf{NOT}\, P\, \mathsf{AND}\, \mathsf{NOT}\, viol(a, P, D))\, \mathsf{UNTIL} \\ \quad (\mathsf{NOT}\, D\, \mathsf{AND}\, P\, \mathsf{AND}\, \mathsf{ALWAYS}\, \mathsf{NOT}\, viol(a, P, D))\big]\end{array}$$

Combining both cases, we get a formula expressing the deadline:

$$\begin{array}{l}\mathcal{M}, s \vDash \mathsf{SOMETIME}\, D\, \mathsf{AND} \\ \quad \big[(\mathsf{NOT}\, D\, \mathsf{AND}\, \mathsf{NOT}\, P\, \mathsf{AND}\, \mathsf{NOT}\, viol(a, P, D))\, \mathsf{UNTIL} \\ \quad ((\mathsf{NOT}\, D\, \mathsf{AND}\, P\, \mathsf{AND}\, \mathsf{ALWAYS}\, \mathsf{NOT}\, viol(a, P, D))\, \mathsf{OR} \\ \quad (D\, \mathsf{AND}\, viol(a, P, D)))\big]\end{array}$$

Which is exactly the LTL-reduction mentioned in proposition 4.6.

The formal reduction, which we explained (informally) above, shows that models expressed by the reduction are also the same as those defined by the semantic definition (thus proving that the reduction is equivalent to the semantic definition):

[1] This assumption is warranted because if P would occur more then once before D, the same restrictions hold on the model.

$\exists t \geq s : \mathcal{M}, t \vDash D$ and
$(\forall s \leq u < t : \mathcal{M}, u \vDash \mathsf{NOT}\, viol(a, P, D))$ and
$((\exists s \leq v < t : \mathcal{M}, v \vDash P$ and $\mathcal{M}, v \vDash \mathsf{ALWAYS\ NOT}\, viol(a, P, D))$ or
$(\forall s \leq w < t : \mathcal{M}, w \nvDash P$ and $\mathcal{M}, t \vDash viol(a, P, D)))$
$\Leftrightarrow$
$\mathcal{M}, s \vDash \mathsf{SOMETIME}\, D$ and
$\mathcal{M}, s \vDash (\mathsf{NOT}\, D\, \mathsf{AND\ NOT}\, viol(a, P, D))\, \mathsf{UNTIL}\, D$ and
$(\exists s \leq v < t : \mathcal{M}, v \vDash \mathsf{NOT}\, D\, \mathsf{AND}\, P\, \mathsf{AND\ ALWAYS\ NOT}\, viol(a, P, D))$ or
$(\forall s \leq w < t : \mathcal{M}, w \vDash \mathsf{NOT}\, P$ and $\mathcal{M}, t \vDash viol(a, P, D)$
$\Leftrightarrow$
$\mathcal{M}, s \vDash \mathsf{SOMETIME}\, D$ and
$\big[\mathcal{M}, s \vDash (\mathsf{NOT}\, D\, \mathsf{NOT}\, P\, \mathsf{AND\ NOT}\, viol(a, P, D))\mathsf{UNTIL}(D\, \mathsf{AND}\, viol(a, P, D))$ or
$(\mathcal{M}, s \vDash (\mathsf{NOT}\, D\, \mathsf{AND\ NOT}\, P\, \mathsf{NOT}\, viol(a, P, D))\, \mathsf{UNTIL}$
$(\mathsf{NOT}\, D\, \mathsf{AND}\, P\, \mathsf{AND\ ALWAYS\ NOT}\, viol(a, P, D)))\big]$
$\Leftrightarrow$
$\mathcal{M}, s \vDash \mathsf{SOMETIME}\, D\, \mathsf{AND}$
$\big[(\mathsf{NOT}\, D\ \mathsf{AND\ NOT}\, P\ \mathsf{AND\ NOT}\, viol(a, P, D))\ \mathsf{UNTIL}$
$((\mathsf{NOT}\, D\ \mathsf{AND}\ P\ \mathsf{AND\ ALWAYS\ NOT}\, viol(a, P, D))\ \mathsf{OR}$
$(D\ \mathsf{AND}\ viol(a, P, D)))\big]$

Reduction of Temporal Prohibitions

We can show that the reduction from the Temporal prohibitions of definition 4.7 to the LTL reduction in proposition 4.9 can be done in a similar manner. In definition 4.7 temporal prohibitions are defined as follows:

$\mathcal{M}, s \vDash \mathsf{FORBIDDEN}(a, P\, \mathsf{BEFORE}\ D) \quad \Leftrightarrow_{def}$
If $\exists t \geq s : \mathcal{M}, t \vDash D$ then
$(((\exists s \leq u < t : \mathcal{M}, u \vDash P$ and $\mathcal{M}, u \vDash \mathsf{NEXT}\ viol(a, P, D))$ and
$(\forall s \leq w < u : \mathcal{M}, w \nvDash P$ and $\mathcal{M}, w \vDash \mathsf{NOT}\ viol(a, P, D)))$ or
$(\forall s \leq v < t : \mathcal{M}, v \nvDash P$ and $\mathcal{M}, v \vDash \mathsf{NOT}\ viol(a, P, D)$ and
$\mathcal{M}, t \vDash \mathsf{ALWAYS\ NOT}\ viol(a, P, D)))$
and if $\forall t \geq s : \mathcal{M}, t \nvDash D$ then $\forall u \geq s :$ if $\mathcal{M}, u \vDash P$ then
$\mathcal{M}, u \vDash \mathsf{NEXT}\ viol(a, P, D)$

This temporal prohibition works, in essence, similar to the deadline specified above. We show (formally) that the semantical definition is equivalent to a formula in LTL, as given by proposition 4.9.

$$
\begin{array}{l}
\textit{If } \exists t \geq s : \mathcal{M}, t \vDash D \text{ then} \\
\quad \big[((\exists s \leq u < t : \mathcal{M}, u \vDash P \text{ and } \mathcal{M}, u \vDash \mathsf{NEXT}\ viol(a,P,D)) \text{ and} \\
\qquad (\forall s \leq w < u : \mathcal{M}, w \nvDash P \text{ and } \mathcal{M}, w \vDash \mathsf{NOT}\ viol(a,P,D))) \text{ or} \\
\quad (\forall s \leq v < t : \mathcal{M}, v \nvDash P \text{ and } \mathcal{M}, v \vDash \mathsf{NOT}\ viol(a,P,D) \text{ and} \\
\qquad \mathcal{M}, t \vDash \mathsf{ALWAYS\ NOT}\ viol(a,P,D))\big] \\
\textit{and if } \forall t \geq s : \mathcal{M}, t \nvDash D \text{ then } \forall u \geq s : \text{ if } \mathcal{M}, u \vDash P \text{ then} \\
\quad \mathcal{M}, u \vDash \mathsf{NEXT}\ viol(a,P,D) \\
\Leftrightarrow \\
\textit{If } \mathcal{M}, s \vDash \mathsf{SOMETIME}\, D \text{ then} \\
\quad \big[((\exists s \leq u < t : \mathcal{M}, u \vDash P\, \mathsf{AND\ NEXT}\ viol(a,P,D)) \text{ and} \\
\qquad (\forall s \leq w < u : \mathcal{M}, w \vDash \mathsf{NOT}\, P\, \mathsf{AND\ NOT}\ viol(a,P,D))) \text{ or} \\
\quad (\mathcal{M}, s \vDash (\mathsf{NOT}\, D\, \mathsf{AND\ NOT}\, P\, \mathsf{AND\ NOT}\ viol(a,P,D))\ \mathsf{UNTIL} \\
\qquad\quad (D\, \mathsf{AND\ ALWAYS\ NOT}\, viol(a,P,D)))\big] \\
\textit{and if } \mathcal{M}, s \vDash \mathsf{ALWAYS\ NOT}\, D \text{ then} \\
\quad \forall u \geq s : \mathcal{M}, u \vDash P\, \mathsf{IMPLIES\ NEXT}\ viol(a,P,D) \\
\Leftrightarrow \\
\textit{If } \mathcal{M}, s \vDash \mathsf{SOMETIME}\, D \text{ then} \\
\quad \big[(\mathcal{M}, s \vDash (\mathsf{NOT\ AND\ NOT\ NOT}\, P\, \mathsf{AND\ NOT}\ viol(a,P,D))\ \mathsf{UNTIL} \\
\qquad\quad (\mathsf{NOT}\, D\, \mathsf{AND}\, P\, \mathsf{AND\ NEXT}\ viol(a,P,D))) \text{ or} \\
\quad (\mathcal{M}, s \vDash (\mathsf{NOT}\, D\, \mathsf{AND\ NOT}\, P\, \mathsf{AND\ NOT}\ viol(a,P,D))\ \mathsf{UNTIL} \\
\qquad\quad (D\, \mathsf{AND\ ALWAYS\ NOT}\, viol(a,P,D)))\big] \\
\textit{and if } \mathcal{M}, s \vDash \mathsf{ALWAYS\ NOT}\, D \text{ then} \\
\quad \forall u \geq s : \mathcal{M}, u \vDash P\, \mathsf{IMPLIES\ NEXT}\ viol(a,P,D) \\
\Leftrightarrow \\
\mathcal{M}, s \vDash \Big(\mathsf{SOMETIME}\, D\ \mathsf{IMPLIES} \\
\qquad \Big[(\mathsf{NOT}\ D\ \mathsf{AND\ NOT}\ P\ \mathsf{AND\ NOT}\ viol(a,P,D))\ \mathsf{UNTIL} \\
\qquad ((D\ \mathsf{AND\ NOT}\ P\ \mathsf{AND\ NEXT\ ALWAYS\ NOT}\ viol(a,P,D))\ \mathsf{OR} \\
\qquad (P\ \mathsf{AND\ NEXT}\ viol(a,P,D)))\Big]\Big)\ \mathsf{AND} \\
\Big(\mathsf{ALWAYS\ NOT}\, D\ \mathsf{IMPLIES\ ALWAYS}\ \Big[P\ \mathsf{IMPLIES\ NEXT}\ viol(a,P,D)\Big]\Big)
\end{array}
$$

Which is exactly expressed in proposition 4.9.

Proofs of Theorems used in Chapter 6

In sections 6.4 and 6.5 several theorems were used to derive the safety and liveness properties of the protocol. Some of these theorems, like theorem 6.13, theorem 6.17 and 6.18 were derived from [Kröger, 1987], and proofs of these theorems can be found there. The other theorems are proven in this section.

Derivation Rule

The following theorem was mentioned in section 6.4 to derive the invariance of a violation predicate:

Theorem A.1 (Derivation Rule) *The following rule is valid:*

$$\frac{\begin{array}{c}\mathcal{M} \models O_x(\alpha < \delta) \\ \mathcal{M} \models \text{at}\,\pi_i \rightsquigarrow \alpha \\ \mathcal{M} \models \text{at}\,\pi_i \rightarrow \neg\Diamond\delta\end{array}}{\mathcal{M} \models \text{at}\,\pi_i \rightarrow \bigcirc\neg viol(x, \alpha, \delta)}$$

Proof. We show semantically that theorem A.1 holds. First, remember that $O(\alpha < \delta)$ stated in means the following (see definition 4.5):

$$\begin{array}{lll}\mathcal{M}, s \models O_x(\alpha < \delta) & \Leftrightarrow & \exists t \geq s : \mathcal{M}, t \models \delta \text{ and} \\ & & (\forall s \leq u < t : \mathcal{M}, u \models \neg V) \text{ and} \\ & & ((\exists s \leq v < t : \mathcal{M}, v \models \alpha \text{ and } \mathcal{M}, v \models \Box\neg V) \text{ or} \\ & & (\forall s \leq w < t : \mathcal{M}, w \not\models \alpha \text{ and } \mathcal{M}, t \models V))\end{array}$$

Now, knowing that $\mathcal{M}, s \models O(\alpha < \delta)$, for s being the state where start_Π holds (remember, we are only interested in those situations where the norms hold and the protocol is started), we also know that there will be a state t such that δ holds. Moreover, all states from s up to t will satisfy $\neg V$, and either there exists a state between s and t (s included) where α holds, in which case $\Box\neg V$ will holds as well, or α does not hold in any state between s and t (again, s included), in which case V holds in t. Suppose we have that, for some state $i \geq s : \mathcal{M}, i \models \text{at}\,\pi_i$. Now we have to prove that $\mathcal{M}, i \models \bigcirc\neg V$. Since we know that $\text{at}\,\pi_i \rightarrow \neg\Diamond\delta$ is true in all states of the computation we know that $\mathcal{M}, i \models \neg\Diamond\delta$, meaning that the moment that $\text{at}\,\pi_i$ holds the deadline has not yet occurred, so we can conclude that $s \leq i < t$. Moreover, since $\text{at}\,\pi_i \rightsquigarrow \alpha$ holds, i.e. $\text{at}\,\pi_i$ counts as α, we know that α happens *before* the deadline. This means that $\exists s \leq i < t : \mathcal{M}, i \models \alpha$ holds, and according to the semantical definition given above we can conclude that $\mathcal{M}, i \models \Box\neg V$ holds as well. Therefore, we can conclude that $\text{at}\,\pi_i \rightarrow \Box\neg V$ and, in particular, $\text{at}\,\pi_i \rightarrow \bigcirc\neg V$ hold.

Liveness Rule

The following theorem was mentioned in section 6.5 to derive the liveness property of a protocol:

Theorem A.2 (Liveness Rule) *The following rule is valid when γ and $\neg\gamma$ are invari-*

ants of Π*:*

$$\mathrm{start}_\Pi \wedge \gamma \rightarrow \Diamond(\mathrm{at}\,\pi_e \wedge \varphi_1) \tag{A.1}$$

$$\mathrm{start}_\Pi \wedge \neg\gamma \rightarrow \Diamond(\mathrm{at}\,\pi_e \wedge \varphi_2) \tag{A.2}$$

$$\overline{\mathrm{start}_\Pi \rightarrow \Diamond(\mathrm{at}\,\pi_e \wedge (\gamma \rightarrow \varphi_1) \wedge (\neg\gamma \rightarrow \varphi_2))} \tag{A.3}$$

Proof. We will only consider the cases where start_Π actually holds, since those are the only cases we are interested in and the rule is trivially true those cases where start_Π does not hold.

Let us assume we can derive (1) and (2), and let us assume that γ actually holds. In this case $\mathrm{start}_\Pi \wedge \gamma$ holds as well and we know, because of (1), that $\Diamond(\mathrm{at}\,\pi_e \wedge \varphi_1)$ holds. Since γ is an invariant of Π, γ holds at all steps of the protocol, and thus at the moment that $\mathrm{at}\,\pi_e$ holds. Therefore, $\Diamond(\mathrm{at}\,\pi_e \wedge \gamma \wedge \varphi_1))$ holds, and we can thus derive that $\Diamond(\mathrm{at}\,\pi_e \wedge (\gamma \rightarrow \varphi_1))$. Since γ holds (and is an invariant of Π), we know that all states satisfy $\neg\gamma \rightarrow \chi$, for arbitrary χ. If we choose $\chi = \varphi$ we thus obtain the desired $\Diamond\pi_e \wedge (\gamma \rightarrow \varphi_1) \wedge (\neg\gamma \rightarrow \varphi_2))$.

Reasoning similarly for $\neg\gamma$ and using (2) instead, we again obtain $\Diamond\pi_e \wedge (\gamma \rightarrow \varphi_1) \wedge (\neg\gamma \rightarrow \varphi_2))$. We can therefore conclude that, whenever start_Π holds, $\Diamond\pi_e \wedge (\gamma \rightarrow \varphi_1) \wedge (\neg\gamma \rightarrow \varphi_2))$ holds, and thus $start_\Pi \rightarrow \Diamond(at\,\alpha_e \wedge (\gamma \rightarrow \varphi_1) \wedge (\neg\gamma \rightarrow \varphi_2))$.

Appendix B

Building Automata from LTL Formulas

Using the procedure described in chapter 5 definition 5.6 (on page 117), we now show how the example automaton A_φ is built from the formula $O(\rho < \delta) \wedge \Box\neg v(\rho,\delta)$.

Given is that:

$$\begin{aligned}
\varphi &= \Diamond\delta \wedge [(\neg\delta \wedge \neg\rho \wedge \neg v(\rho,\delta) \textbf{ until} \\
&\quad ((\rho \wedge \neg\delta \wedge \Box\neg v(\rho,\delta) \vee (\neg\rho \wedge \delta \wedge v(\rho,\delta)))] \wedge \Box\neg v(\rho,\delta) \\
P &= \{\rho, \delta, v(\rho,\delta)\} \\
\psi &= [(\neg\delta \wedge \neg\rho \wedge \neg v(\rho,\delta)) \textbf{ until} \\
&\quad ((\rho \wedge \neg\delta \wedge \Box\neg v(\rho,\delta) \vee (\neg\rho \wedge \delta \wedge v(\rho,\delta)))] \wedge \Box\neg v(\rho,\delta) \\
\omega &= ((\rho \wedge \neg\delta \wedge \Box\neg v(\rho,\delta) \vee (\neg\rho \wedge \delta \wedge v(\rho,\delta)))] \wedge \Box\neg v(\rho,\delta) \\
cl(\varphi) &= \{\{\varphi, \Diamond\delta, \psi, \Box\neg v(\rho,\delta), \delta, \neg v(\rho,\delta), \neg\delta \wedge \neg\rho \wedge \neg v(\rho,\delta), \omega, \\
&\quad \neg\rho, \rho \wedge \neg\delta \wedge \Box\neg v(\rho,\delta), \neg\rho \wedge \delta \wedge v(\rho,\delta), \rho, v(\rho,\delta), \textbf{true}, \textbf{false}\}\} \\
\Sigma &= 2^P \\
&= \{\{\rho, \delta, v(\rho,\delta)\}, \{\rho,\delta\}, \{\rho, v(\rho,\delta)\}, \{\delta, v(\rho,\delta)\}, \{\rho\}, \{\delta\}, \{v(\rho,\delta)\}, \varnothing\}
\end{aligned}$$

For readability we will use v instead of $v(\rho,\delta)$.

Now we start the procedure:

$$\begin{aligned}
S &:= \varnothing\\
\Delta &:= \varnothing\\
\mathit{sat}(\{\{\varphi\}\}) &= \{\{\varphi, \Diamond\delta, \psi, \Box\neg\upsilon, \bigcirc\Diamond\delta, \neg\upsilon, \bigcirc\Box\neg\upsilon, \neg\delta\wedge\neg\rho\wedge\neg\upsilon, \neg\delta, \neg\rho, \bigcirc\psi\},\\
&\qquad \{\varphi, \Diamond\delta, \psi, \Box\neg\upsilon, \bigcirc\Diamond\delta, \neg\upsilon, \bigcirc\Box\upsilon, \omega, \rho\wedge\neg\delta\wedge\Box\neg\upsilon, \rho, \neg\delta\}\}\\
S_0 &:= \{\{\varphi, \Diamond\delta, \psi, \Box\neg\upsilon, \neg\upsilon, \bigcirc\Box\neg\upsilon, \neg\delta\wedge\neg\rho\wedge\neg\upsilon, \neg\delta, \neg\rho\}, && (s_1)\\
&\qquad \{\varphi, \Diamond\delta, \psi, \Box\neg\upsilon, \neg\upsilon, \omega, \rho\wedge\neg\delta\wedge\Box\neg\upsilon, \rho, \neg\delta\}\} && (s_2)\\
\mathit{unp} &:= \{(s_1, \{\Diamond\delta, \Box\neg\upsilon, \psi\}), (s_2, \{\Diamond\delta, \Box\neg\upsilon\})\}
\end{aligned}$$

We first select $(s_1, \{\Diamond\delta, \Box\neg\upsilon, \psi\})$:

$$\begin{aligned}
\mathit{sat}(\{\{\Diamond\delta, \Box\neg\upsilon, \psi\}\}) &= \{\{\Diamond\delta, \psi, \Box\neg\upsilon, \bigcirc\Diamond\delta, \neg\upsilon, \bigcirc\Box\neg\upsilon, \omega, \rho\wedge\neg\delta\wedge\Box\neg\upsilon, \rho, \neg\delta\},\\
&\qquad \{\Diamond\delta, \psi, \Box\neg\upsilon, \bigcirc\Diamond\delta, \neg\upsilon, \bigcirc\Box\neg\upsilon, \neg\delta\wedge\neg\rho\wedge\neg\upsilon, \bigcirc\psi, \neg\delta, \neg\rho\}\}\\
P(s_1) &= \varnothing\\
nP(s_1) &= \{\upsilon\}\\
\Delta &:= \Delta\cup\{(s_1, \varnothing, s_1), (s_1, \varnothing, s_2)\}\\
\mathit{unp} &:= \{(s_2, \{\Diamond\delta, \Box\neg\upsilon\})\}
\end{aligned}$$

The transitions from s_1 to itself, and s_1 to s_2 can be seen from the results of the *saturate*, where the first set is the same set (intersected with $cl(\varphi)$) as s_2 with the exception of φ, and the second set is the same set (intersected with $cl(\varphi)$) as s_1, again with the exception of φ itself. The equivalence between the first set (lets call it s_2') and s_2, and the equivalence between the second set (lets call it s_1') and s_1, follow from the following equivalence:

$$\{\varphi_1, \varphi_2, \varphi_1\wedge\varphi_2\} \Leftrightarrow \{\varphi_1, \varphi_2\}$$

Therefore, $s_1' \Leftrightarrow s_1$ and $s_2' \Leftrightarrow s_2$.

Next we look at $(s_2, \{\Diamond\delta, \Box\neg\upsilon\})$:

$$\begin{aligned}
\mathit{sat}(\{\{\Diamond\delta, \Box\neg\upsilon\}\}) &= \{\{\Diamond\delta, \Box\neg\upsilon, \bigcirc\Diamond\delta, \neg\upsilon, \bigcirc\Box\neg\upsilon\},\\
&\qquad \{\Diamond\delta, \Box\neg\upsilon, \delta, \neg\upsilon, \bigcirc\Box\neg\upsilon\}\}\\
P(s_2) &= \{\rho\}\\
nP(s_2) &= \{\upsilon, \delta\}\\
\Delta &:= \Delta\cup\{(s_2, \{\rho\}, s_3), (s_2, \{\rho\}, s_4)\}\\
s_3 &= \{\Diamond\delta, \Box\neg\upsilon, \neg\upsilon\}\\
s_4 &= \{\Diamond\delta, \Box\neg\upsilon, \delta, \neg\upsilon\}\\
\mathit{unp} &:= \{(s_3, \{\Diamond\delta, \Box\neg\upsilon\}), (s_4, \{\Box\neg\upsilon\})\}
\end{aligned}$$

Next we look at $(s_3, \{\Diamond\delta, \Box\neg v\})$

$$\begin{aligned}
sat(\{\{\Diamond\delta, \Box\neg v\}\}) &= \{\{\Diamond\delta, \Box\neg v, \bigcirc\Diamond\delta, \neg v, \bigcirc\Box\neg v\}, \\
&\quad\ \{\Diamond\delta, \Box\neg v, \delta, \neg v, \bigcirc\Box\neg v\}\} \\
P(s_3) &= \varnothing \\
nP(s_3) &= \{v\} \\
\Delta &:= \Delta \cup \{(s_3, \{\rho\}, s_3), (s_3, \{\delta\}, s_3), (s_3, \{\rho, \delta\}, s_3), \\
&\quad\ (s_3, \{\rho\}, s_4), (s_3, \{\delta\}, s_4), (s_3, \{\rho, \delta\}, s_4)\} \\
unp &:= \{(s_4, \{\Box\neg v\})\}
\end{aligned}$$

Next we look at $(s_4, \{\Box\neg v\})$:

$$\begin{aligned}
sat(\{\Box\neg v\}) &= \{\{\Box\neg v, \neg v, \bigcirc\Box\neg v\}\} \\
P(s_4) &= \{\delta\} \\
nP(s_4) &= \{v\} \\
\Delta &:= \Delta \cup \{(s_4, \{\delta\}, s_5), (s_4, \{\delta, \rho\}, s_5)\} \\
s_5 &= \{\Box\neg v\} \\
unp &:= \{(s_5, \{\Box\neg v\})\}
\end{aligned}$$

Finally, we look at $(s_5, \{\Box\neg v\})$:

$$\begin{aligned}
sat(\{\Box\neg v\}) &= \{\{\Box\neg v, \neg v, \bigcirc\Box\neg v\}\} \\
P(s_5) &= \varnothing \\
nP(s_4) &= \{v\} \\
\Delta &:= \Delta \cup \{(s_5, \rho, s_5), (s_5, \{\delta\}, s_5), (s_4, \{\delta, \rho\}, s_5), (s_5, \varnothing, s_5)\} \\
unp &:= \varnothing
\end{aligned}$$

This gives us the automaton A_φ as presented in figure 5.3 on page 119, with the acceptance condition (in accordance with the definition on page 116) $\mathcal{F}_\varphi = \{\{s_4, s_5\}, \{s_2, s_4, s_5\}\}$.

Appendix C

Organ Donation: Example Protocol & Norms

In chapter 6 we used a real-world protocol for clarifying the formal verification of the norm compliance of protocols. The example we used is an model-protocol used for making the decision whether the organs of a recently deceased patient can be extracted for transplantation purposes. Here we give the full formalised version of this example protocol, and the norms (with their formalisation) that govern this domain.

Formalised Protocol

The protocol from figure 6.1 in section 6.1 (on page 136) can be formalised into the following, using the language mentioned in section 6.3.1:

```
Initial $know\_permission(d, remove\_organ) := \texttt{FALSE}$,
$know\_potential\_donor(d, y) := \texttt{FALSE}$,
$know\_not\_permission(d, remove\_organ) := \texttt{FALSE}$,
$know\_not\_potential\_donor(d, y) := \texttt{FALSE}$;
cobegin $\Pi$

$\Pi =$
    $\pi_0$ : $\langle$check_criteria_&_contra-indications$\rangle$;
    $\pi_1$ : if $know\_criteria(d, y) \wedge know\_no\_contra\text{-}indication(d, y)$
          then $\pi_2$ : $know\_potential\_donor(d, y) := \texttt{TRUE}$
          else $\pi_3$ : $know\_not\_potential\_donor(d, y) := \texttt{TRUE}$
          fi;
    $\pi_4$ : if $know\_potential\_donor(d, y)$
```

```
        then $\pi_5$ : $\langle$consult_donor_register$\rangle$
        fi;
$\pi_6$ : if $know\_potential\_donor(d,y) \wedge know\_registered(d,y)$
        then $\pi_7$ : $\langle$check_permission$\rangle$
        else $\pi_8$ : if $know\_potential\_donor(d,y) \wedge know\_not\_registered(d,y)$
                  then $\pi_9$ : $\langle$check_other_statement$\rangle$
                  fi
        fi;
$\pi_{10}$ : if $know\_potential\_donor(d,y) \wedge know\_registered(d,y) \wedge$
           $know\_not\_statement\_permission(d,y)$
        then $\pi_{11}$ : $know\_potential\_donor(d,y)$ := FALSE;
             $\pi_{12}$ : $know\_not\_potential\_donor(d,y)$ := TRUE
        else $\pi_{13}$ : if $know\_potential\_donor(d,y) \wedge$
                  $(know\_registered(d,y) \wedge know\_statement\_permission(d,y)) \vee$
                  $(know\_not\_registered(d,y) \wedge know\_other\_statement(d,y))$
                  then $\pi_{14}$ $\langle$inform_relatives$\rangle$
                  fi
        fi;
$\pi_{15}$ : if $(know\_potential\_donor(d,y) \wedge$
           $(know\_not\_registered(d,y) \wedge know\_not\_other\_statement(d,y)) \vee$
           $(know\_registered(d,y) \wedge know\_includes\_escape\_clause(z)))$
        then $\pi_{16}$ : $\langle$ask_relatives$\rangle$;
                  $\pi_{17}$ : if $know\_not\_relative\_permission(d,y)$
                        then $\pi_{18}$ : $know\_potential\_donor(d,y)$ := FALSE;
                             $\pi_{19}$ : $know\_not\_potential\_donor(d,y)$ := TRUE
                  fi
        fi;
$\pi_{20}$ : if $know\_potential\_donor(d,y) \wedge know\_natural\_death(d,y)$
        then $\pi_{21}$ : $\langle$report_to_transplant_coordinator$\rangle$
        else $\pi_{22}$ : if $know\_potential\_donor(d,y) \wedge know\_non\_natural\_death(d,y))$
                  then $\pi_{23}$ : $\langle$report_to_coroner$\rangle$;
                       $\pi_{24}$ : $\langle$ask_DA_for_permission$\rangle$;
                       $\pi_{25}$ : if $know\_DA\_permission(d,y)$
                       then $\pi_{26}$ : $\langle$report_to_transplant_coordinator$\rangle$
                       else $\pi_{27}$ : $know\_potential\_donor(d,y)$ := FALSE;
                            $\pi_{28}$ : $know\_not\_potential\_donor(d,y)$ := TRUE
                       fi
                  fi
        fi;
$\pi_{29}$ : if $know\_potential\_donor(d,y)$
        then $\pi_{30}$ : $know\_permission(d, remove\_organ)$ := TRUE
        else $\pi_{31}$ : if $know\_not\_potential\_donor(d,y)$
                  then $\pi_{32}$ : $know\_not\_permission(d, remove\_organ)$ := TRUE
                  fi
        fi;
$\pi_{33}$ : ⟨fill_donor_form⟩;
$\pi_{34}$ : stop
```

Applicable Norms

Using the logical formalism presented in chapter 6.3 we can translate the following norms into a formal representation that we used in section 6.4 and section 6.5 to verify the safety and liveness properties, respectively, of the formal protocol presented above.

Article 10

3. The register can be consulted by or under the authority of a doctor during day and night whenever that is necessary considering the intended removal of an organ.

$$P_x((DO_x\ consult(x, register)) \mid (under\_authority(d, consult(x, register)) \wedge necessary\_for(consult(x, register), intended(organ\_removal))))$$

Article 14

1. Before an organ is removed, death is certified by a doctor who may not be involved in the removal or transplantation of the organ. If the intention exists to remove the organ from an insufflated body, death is certified by means of the latest state of medically valid methods and criteria for the determination of brain death by a very knowledgeable doctor. The manner of determining the brain death is recorded in a statement of which a model is included the in article 15, first item, mentioned protocol.

$$O_d((DO_d\ certify\_death(d, y)) < (DO_{d'}\ remove\_organ(d', y))) \wedge F_d((DO_d\ remove\_organ(d, y)) \mid (DONE_d\ certify\_death(d, y)))$$

Article 16

At the removal of an organ from a body, the autopsy as meant by article 3 of the Law on disposal of the dead (Stb. 1991, 133) is not carried out by the doctor who is involved in the removal or implantation of the organ.

$$F_d((DO_d\ autopsy(d, y)) \mid (DONE_d\ remove\_organ(d, y)))$$

Article 17

An organ may not be removed when suspicion of a non-natural death exists, until it has been established that the district attorney grants the permission as mentioned in article 76, second item, of the Law on disposal of the dead.

$$\begin{aligned}&F_d((DO_d\ remove\_organ(d,y))\ <\\&\quad(DONE_u\ grants(u, permission, organ\_removal))\ |\\&\quad(suspicion(non\_natural\_death, y)))\end{aligned}$$

Article 18

1. The doctor who determines the death makes sure that organs supposedly coming available for implantation are announced to an organ centre.

$$\begin{aligned}&O_d((DO_d\ announce(d, organ(p), c)) > available(organ(y))\ |\\&\quad(DONE_d\ certify\_death(d,y)))\end{aligned}$$

Article 20

1. The one who determines the death takes care of checking whether a statement of intent as mentioned by article 9 and article 10 is present. If a statement of intent emerging from a register correspond to another existing statement as meant by article 9 the last made statement is applicable.

$$O_d((DO\ check(d,z))\ >\ (DONE\ certify\_death(d,y)))$$

2. If no statement of intent mentioned by article 9 and article 10 is present the doctor mentioned in the first item takes, in accordance with the protocol, care of consulting the person or persons mentioned by article 11.

$$\begin{aligned}&O_d((DO_d\, confer\_with(d,x) > (\Diamond DONE_d\, check(d,z)))\ |\\&\quad((\neg statement\_of\_intent(w,y) \wedge relative(x) \wedge w = z) \wedge\\&\quad\Diamond DONE_d\, certify\_death(d,y)))\end{aligned}$$

References

C. Alchourón and E. Bulygin. *Normative Systems*. Springer-Verlag, 1971.

H. Aldewereld, F. Dignum, J.-J. Meyer, and J. Vázquez-Salceda. Proving norm compliancy of protocols in electronic institutions. Technical Report UU-CS-2005-010, Institute of Information and Computing Sciences, Utrecht University, 2005.

H. Aldewereld, A. García-Camino, F. Dignum, P. Noriega, J.-A. Rodríguez-Aguilar, and C. Sierra. Operationalisation of norms for usage in electronic institutions. In *Proceedings of the Fifth International Conference on Autonomous Agents and Multiagent Systems (AAMAS'06)*. Hakodate, Japan, 2006a.

H. Aldewereld, A. García-Camino, F. Dignum, P. Noriega, J.-A. Rodríguez-Aguilar, and C. Sierra. Operationalisation of norms for usage in electronic institutions. In *Proceedings of the AAMAS06 Workshop on Coordination, Organization, Institutions and Norms in agent systems (COIN@AAMAS'06)*. Hakodate, Japan, 2006b.

A. Anderson. A reduction of deontic logic to alethic modal logic. *Mind*, 67:pages 100–103, 1958.

G. Antoniou. *Nonmonotonic Reasoning*. MIT Press, Cambridge, MA, 1997.

N. Belnap and M. Perloff. Seeing to it that: a canonical form for agentives. *Theoria*, 54:pages 175–199, 1988.

T. Bench-Capon and F. Coenen. Isomorphism and legal knowledge based systems. *Artificial Intelligence and Law*, 1:pages 65–86, 1992.

J. Boella and L. Lesmo. Deliberative normative agents. In C. Dellarocas and R. Conte, editors, *Proceedings of the Workshop on Norms and Institutions in Multi-Agent Systems*, ACM-AAAI, pages 15–25. ACM Press, 2000.

J. Broersen, F. Dignum, V. Dignum, and J.-J. Ch. Meyer. Designing a deontic logic of deadlines. In *7th International Workshop on Deontic Logic in Computer Science (DEON'04)*. Portugal, 2004.

J.R. Büchi. On a decision method in restricted second order arithmetic. In *Proceedings of the 1960 International Conference on Logic, Methodology and Philosophy of Science*, pages 1–11. Stanford University Press, Stanford, CA, 1962.

C. Castelfranchi, F. Dignum, C. Jonker, and J. Treur. Deliberative normative agents: Principles and architectures. In N. Jennings and Y. Lesperance, editors, *ATAL '99*, volume 1757 of *LNAI*, pages 364–378. Springer Verlag, Berlin Heidelberg, 2000.

P. Cohen and H. Levesque. Teamwork. *Nous*, 25(4):pages 487–512, 1991.

J. Coleman. *Foundations of Social Theory*. Belknap Press of Harvard University Press Cambridge, Massachusetts, 1990.

M. Dastani, B. van Riemsdijk, J. Hulstijn, F. Dignum, and J.-J. Ch. Meyer. Enacting and deacting roles in agent programming. In *Proceedings of the 5th International Workshop on Agent-Oriented Software Engineering (AOSE'04)*, volume 3382 of *LNCS*, pages 189–204. Springer-Verlag, New York, 2004.

M. Dastani, B. van Riemsdijk, and J.-J. Ch. Meyer. Programming multi-agent systems in 3apl. In R. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, Berlin, 2005.

C. De Vey Mestdagh. *Rekentuig of Rekenmeester?*. Ph.D. thesis, Rijksuniversiteit Groningen, 1997.

F. Dignum. Autonomous agents with norms. *AI and Law*, 7:pages 69–79, 1999.

F. Dignum. Abstract norms and electronic institutions. In *Proceedings of the International Workshop on Regulated Agent-Based Social Systems: Theories and Applications (RASTA '02), Bologna*, pages 93–104. 2002.

F. Dignum, J. Broersen, V. Dignum, and J.-J. Ch. Meyer. Meeting the deadline: Why, when and how. In *3rd Goddard Workshop on Formal Approaches to Agent-Based Systems (FAABS)*. Maryland, 2004.

F. Dignum, D. Kinny, and L. Sonenberg. From desires, obligations and norms to goals. *Cognitive Science Quarterly*, 2(3-4):pages 407–430, 2002a.

F. Dignum, D. Morley, and E. Sonenberg. Towards socially sophisticated bdi agents. In *Proceedings of DEXA Workshop*, pages 1134–1140. 2000.

V. Dignum. *A Model for Organizational Interaction: Based on Agents, Founded in Logic*. Ph.D. thesis, Universiteit Utrecht, The Netherlands, 2004.

V. Dignum and F. Dignum. Modeling agent societies: Coordination frameworks and institutions. In P. Brazdil and A. Jorge, editors, *Progress in Artificial Intelligence*, volume 2258 of *LNAI*, pages 191–204. 2001.

V. Dignum, J.-J. Ch. Meyer, F. Dignum, and H. Weigand. Formal specification of interaction in agent societies. In *2nd Goddard Workshop on Formal Approaches to Agent-Based Systems (FAABS)*. Maryland, 2002b.

J. van Eck. A system of temporally relative modal and deontic predicate logic and its philosophical applications. *Logique et Analyse*, 100:pages 249–381, 1982.

M.H. van Emden and R.A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM (JACM)*, 23(4):pages 733–742, 1976.

T. van Engers and G. van der Veer. Power: an illustration of task based design for groupware. In R. Traunmülla and A. Rizzo, editors, *Proceeding of the Workshop on Distributed Cognition and Distributed Knowledge: Key Issues in Design for e-Commerce and e-Government*. 2000.

M. Esteva. *Electronic Institutions: from specification to development*. Ph.D. thesis, Universitat Politéchnica de Catalunya, 2003.

M. Esteva, J. Padget, and C. Sierra. Formalizing a language for institutions and norms. In J.-J. Ch. Meyer and M. Tambe, editors, *Intelligent Agents VIII*, LNAI 2333, pages 348–366. Springer Verlag, 2001.

M. Esteva, J. Rodríguez-Aguilar, B. Rosell, and J. Arcos. Ameli: An agent-based middleware for electronic institutions. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multi-agent Systems*. New York, US, 2004a.

M. Esteva, W. Vasconcelos, C. Sierra, and J. Rodríguez-Aguilar. Verifying norm consistency in electronic institutions. In *Proceedings of The AAAI-04 Workshop on Agent Organizations: Theory and Practice (ATOP)*. San Jose, California, 2004b.

R. Fikes and N. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3/4):pages 189–208, 1971.

N. Findler and R. Malyankar. Social structures and the problem of coordination in intelligent agent societies. In *Invited talk at the Special Session on Agent-Based Simulation, Planning and Control in ICMAS World Congress 2000*. Laussanne, Switzerland, 2000.

J. Fox and S. Das. *Safe and Sound*. AAAIPress/MIT Press, 1^st edition, 1999.

D. Gabbay. *Investigations in Modal and Tense Logics with Applications to Problems in Philosophy and Linguistics*. Reidel, Dordrecht / Boston, 1976.

D. Gambetta. *Can We Trust Trust?*, pages 213–237. Basil Blackwell, Oxford, 1990.

A. García-Camino and J.-A. Rodríguez-Aguilar. A translator for norms to operational constraints. 2006. Unpublished work.

A. García-Camino and J. Rodríguez-Aguillar. Implementing norms in electronic institutions. In *Proceedings of the 4th Int. Joint Conf. on Autonomous Agents & Multi Agent Systems (AAMAS-05)*. Utrecht, The Netherlands, 2005.

A. García-Camino, J. Rodríguez-Aguillar, C. Sierra, and W. Vasconcelos. A distributed architecture for norm-aware agent societies. In *Proc. of the 3rd Int. Workshop on Declarative Agent Languages and Technologies (DALT 2005)*. Utrecht, The Netherlands, 2005.

D. Grossi. Personal discussion on ontological commitments of norm formalisms. 2004.

D. Grossi, H. Aldewereld, and F. Dignum. *Ubi Lex, Ibi Poena*: Designing norm enforcement in e-institutions. In V. Dignum and J. Vázquez-Salceda, editors, *Proceedings of the AAMAS Workshop on Coordination, Organisation, Institutions and Norms in agent systems (COIN)*. Hakodate Japan, 2006a.

D. Grossi, H. Aldewereld, J. Vázquez-Salceda, and F. Dignum. Ontological aspects of the implementation of norms in agent-based electronic institutions. *Journal of Computational and Mathematical Organization Theory*, 12(2-3):pages 251–275, 2006b. Special Issue about Normative Multiagent Systems.

D. Grossi, F. Dignum, and J.-J. Ch. Meyer. Contextual taxonomies. In J. Leite and P. Toroni, editors, *Proceedings of CLIMA V Workshop, Lisbon, September*, pages 2–17. 2004.

D. Grossi, J.-J. Ch. Meyer, and F. Dignum. Classifactory aspects of counts-as: An analysis in modal logic. *Journal of Logic and Computation*, 16(5):pages 613–643, 2006c.

D. Grossi, J.-J. Ch. Meyer, and F. Dignum. Counts-as: Classification or constitution? an answer using modal logic. In L. Goble and J.-J. Ch. Meyer, editors, *Deontic Logic and Artificial Normative Systems, Eighth International Workshop on Deontic Logic in Computer Science (DEON'06)*, volume 4048 of *LNAI*, pages 115–130. Springer, 2006d.

B. Grosz and S. Kraus. Collaborative plans for complex group actions. *Artificial Intelligence*, 86(2):pages 269–358, 1996.

T. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):pages 199–220, 1993.

J. Hage. Dialectical models in artificial intelligence and law. *Artificial Intelligence and Law*, 8(2-3):pages 137–172, 2000.

H. Hart. *The concept of law*. Clarendon Press Oxford, 1961.

J. Hintikka. *Knowledge and Belief: an introduciton to the logic of two notions*. Cornell University Press, 1962.

G. Holzmann. The model checker spin. *IEEE Transactions on Software Engineering*, 23(5):pages 279–295, 1997.

A. Hommersom, P. Lucas, and M. Balser. Meta-level verification of the quality of medical guidelines using interactive theorem proving. In *Proceedings of JELIA-2004*, number 3229 in LNAI, pages 654–666. Springer, Berlin Heidelberg, 2004.

J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison Wesley, 3rd edition, 2006.

N. Jennings. Commitment and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, 8(3):pages 223–250, 1993.

N. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using join intentions. *Artificial Intelligence*, 75(2):pages 195–240, 1995.

A. Jones. Towards a formal theory of defeasible deontic conditionals. *Annals of Mathematics and Artificial Intelligence*, 9(1-2):pages 151–166, 1993.

A. Jones and M. Sergot. Deontic logic in the representation of law: towards a methodology. *Artificial Intelligence and Law*, 1:pages 45–64, 1992.

A. Jones and M. Sergot. A formal characterisation of institutional power. *Journal of the IGPL*, 4(3):pages 429–445, 1996.

R. van Kralingen. *Frame-Based Conceptual Models of Statute Law.* Kluwer Law Int, 1995.

F. Kröger. *Temporal Logic of Programs*, volume 8 of *EACTS monographs on theoretical computer science.* Springer-Verlag, 1987.

S. Kumar, M. J. Huber, P. R. Cohen, and D. McGee. Toward a formalism for conversation protocols using joint intention theory. *Computational Intelligence*, 18(2):pages 174–228, 2002.

V. Lifschitz. Circumscription. In D. Gabbay, C. Hogger, and J. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, pages 297–352. Clarendon Press, Oxford, 1994.

F. López y Lopez. *Social Power and Norms: Impact on Agent Behaviour.* Ph.D. thesis, Faculty of Engineering and Applied Science, Univ. of Southampton, 1997.

F. López y Lopez and M. Luck. Towards a model of the dynamics of normative multi-agent systems. In G. Lindemann, D. Moldt, M. Paolucci, and B. Yu, editors, *Proceedings of the International Workshop on Regulated Agent-Based Social Systems: Theories and Applications (RASTA '02)*, volume 318 of *Mitteilung*, pages 175–194. Universität Hamburg, 2002.

F. López y Lopez, M. Luck, and M. d'Inverno. A framework for norm-based inter-agent dependence. In *Proceedings of the 3rd Mexican International Conference on Computer Science*, pages 31–40. SMCC-INEGI, 2001.

W. Lukaszewicz. *Non-Monotonic Reasoning, Formalization of Commonsense Reasoning.* Ellis Horwood, New York, 1990.

J. McCarthy. Circumscription: A form of non-monotonic reasoning. *Artificial Intelligence*, 13(1-2):pages 27–39, 1980.

L. McCarthy. Ownership: a case study in the representation of legal concepts. *Artificial Intelligence and Law*, 10:pages 135–161, 2002.

J.-J. Ch.. Meyer. A different approach to deontic logic. *Notre Dame Journal of Formal Logic*, 29(1):pages 109–136, 1988.

J.-J. Ch. Meyer and W. van der Hoek. *Epistemic Logic for AI and Computer Science.* Cambridge University Press, New York, USA, 1995.

J.-J. Ch. Meyer, H. Weigand, and R. Wieringa. A specification language for static, dynamic and deontic integrity constraints. In J. Demetrovics and B. Thalheim, editors, *2nd Symposium on Mathematical Fundamentals of Database Systems (MFDBS 89)*, volume 364 of *Lecture Notes in Computer Science*, pages 347–366. Springer, 1989.

J.-J. Ch. Meyer and R. J. Wieringa, editors. *Deontic Logic in Computer Science*. Wiley Professional Computing, England, 1993.

J.-J. Ch. Meyer, R. J. Wieringa, and F. P. M. Dignum. The role of deontic logic in the specification of information systems. Technical Report UU-CS-1996-55, Universiteit Utrecht, The Netherlands, 1996.

J.-J. Ch. Meyer, R. J. Wieringa, and F. P. M. Dignum. The role of deontic logic in the specification of information systems. *Kluwer International Series In Engineering And Computer Science*, pages 71–115, 1998.

Y. Moses and M. Tennenholtz. Artificial social systems. *Computers and AI*, 14(6):pages 533–562, 1995.

P. Noriega. *Agent-Mediated Auctions: The Fishmarket Metaphor*. Ph.D. thesis, Inst. d'Investigació en Intel.ligència Artificial, 1997.

D. North. *Institutions, Institutional Change and Economic Performance*. Cambridge University Press, 1^{st} edition, 1990.

A. Omicini, A. Ricci, M.Viroli, C. Castelfranchi, and L. Tummolini. Coordination artifacts: Environment-based coordination for intelligent agents. In N. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *Proceedings of the third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004)*, pages 286–293. New York, USA, 2004.

P. Pasquier, R. A. Flores, and B. Chaib-draa. Modeling flexible social commitments and their enforcement. In unknown, editor, *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'06)*, volume 1, pages 1372–1375. ACM Press, Hakodate, Japan, 2006a.

P. Pasquier, R. A. Flores, and B. Chaib-draa. An ontology of social control tools. In unknown, editor, *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'06)*, volume 1, pages 1369–1372. ACM Press, Hakodate, Japan, 2006b.

A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science*, pages 46–57. IEEE, Providence, RI, New York, 1977.

J. Pollock. *Defeasible Reasoning*. Cognitive Science, 1987.

H. Prakken. An argumentation framework in default logic. *Annals of Mathematics and Artificial Intelligence*, 9(1-2):pages 93–132, 1993.

H. Prakken. Two approaches to defeasible reasoning. In A. Jones and M. Sergot, editors, *Proceedings of the International Workshop on Deontic Logic in Computer Science (DEON'94)*, pages 281–295. Tano A.S., Oslo, 1994.

H. Prakken. *Logical Tools for Modeling Legal Argument. A Study of Defeasible Reasoning in Law.* Law and Philosophy Library. Kluwer Academic Publishers, Dordrecht, 1997.

H. Prakken. Modelling defeasibility in law: Logic or procedure? *Fundamenta Informaticae*, 48(2-3):pages 253–271, 2001a.

H. Prakken. Modelling reasoning about evidence in legal procedure. In *Proceedings of the Eighth International Conference on Artificial Intelligence and Law, ICAIL 2001*, pages 119–128. ACM Press, 2001b.

H. Prakken and G. Sartor. Argument-based logic programming with defeasible priotities. *Journal of Applied Non-classical Logics*, 7:pages 25–75, 1997a.

H. Prakken and G. Sartor. *Logical models of legal argumentation.* Kluwer, Dordrecht, 1997b.

H. Prakken and G. Sartor. The role of logic in computational models of legal argument: A critical survey. In *Computational Logic: Logic Porgramming and Beyond*, LNAI 2048, pages 342–381. Springer-Verlag, 2002.

H. Prakken and M. Sergot. Contrary-to-duty obligations. *Studia Logica*, 57(1):pages 91–115, 1996.

E. Rissland, K. Ashley, and R. Loui. Ai and law: a fruitful synergy. *Artificial Intelligence*, 150:pages 1–15, 2003.

J. Rodriguez. *On the Design and Construction of Agent-mediated Electronic Institutions.* Ph.D. thesis, Inst. d'Investigació en Intel.ligència Artificial, 2001.

W. Scott. *Institutions and Organizations.* Sage, 2nd edition, 2001.

M.J. Sergot, F. Sadri, R.A. Kowalski, and F. Kriwaczek. The british nationality act as a logic program. *Communications of the ACM*, 29(5):pages 370–386, 1986.

M. Sipser. *Introduction to the Theory of Computation.* PWS Publishing Company, Boston, 1997.

I. Smith, P. Cohen, J. Bradshaw, M. Greaves, and H. Holmback. Designing conversation policies using joint intention theory. In *Proceedings of the Third International Conference on Mulit Agent Systems (ICMAS-98)*, pages 269–276. IEEE Press, Paris, 1998.

L. Staiger. ω-languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 339–387. Springer-Verlag, Berlin, 1997.

M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:pages 83–124, 1997.

Y.-H. Tan, A. Mitrakas, and W. Thoen. *A formal analysis of Incoterms for electronic commerce*, volume 12 of *ITER*. Kluwer, Deventer, 1998.

Dutch DPA. The police files act [wet politieregisters (wpolr)]. Available from the website of the Dutch Data Protection Authority (College Bescherming Persoonsgegevens): `http://www.dutchdpa.nl/`, 2006.

Dutch Transplant Association. Model-protocol postmortal organ and tissue donation [modelprotocol postmortale orgaan- en weefseldonatie]. Available from the website of the Dutch Transplant Association (Nederlandse Transplantatie Stichting): `http://www.transplantatiestichting.nl/`, 2006.

W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Methods and Semantics, pages 133–192. Elsevier Science Publishers, Amsterdam, 1990.

J. Tomberlin. Contrary-to-duty imperatives and conditional obligation. *Noûs*, 15(3):pages 357–375, 1981.

L. van der Torre. Violated obligations in a defeasible deontic logic. In A. Cohn, editor, *Proceedings of the Europian Conference on Artificial Intelligence (ECAI'94)*, pages 371–375. John Wiley & Sons Ltd., Chicester, 1994.

R. Tuomela. *The Importance of Us: A Phylosophical Study of Basic Social Norms*. Standford University Press, 1995.

M. Vardi. An automata-theoretic approach to linear temporal logic. *Logics for Concurrency: Structure versus Automata*, pages 238–265, 1996.

M. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):pages 1–37, 1994.

J. Vázquez-Salceda. *The Role of Norms and Electronic Institutions in Multi-Agent Systems. The HARMONIA framework*. Whitestein Series in Software Agent Technology. Birkhäuser Verlag, 2004.

J. Vázquez-Salceda, H. Aldewereld, and F. Dignum. Implementing norms in multiagent systems. In G. Lindemann, J. Denzinger, I. Timm, and R. Unland, editors, *Multiagent System Technologies*, LNAI 3187, pages 313–327. Springer-Verlag, 2004.

J. Vázquez-Salceda, H. Aldewereld, and F. Dignum. Norms in multiagent systems: From theory to practice. *International Journal of Computer Systems Science & Engineering*, 20(4):pages 225–236, 2005.

J. Vázquez-Salceda, U. Cortés, and J. Padget. Integrating the organ and tissue allocation processes through an agent-mediated electronic institution. In M. T. Escrig, F. Toledo, and E. Golobardes, editors, *Topics in Artificial Intelligence: 5th Catalonian Conference on AI*, number 2504 in LNAI, pages 309–321. Springer-Verlag, 2002.

J. Vázquez-Salceda, V. Dignum, and F. Dignum. Organizing multiagent systems. Technical Report UU-CS-2004-015, Institute of Information and Computing Sciences, Utrecht University, 2004.

H. Verhagen. *Norm Autonomous Agents*. Ph.D. thesis, Stockholm University, 2000.

B. Verheij. *Rules, Reasons, Arguments: Formal studies of argumentation and defeat*. Ph.D. thesis, Universiteit Maastricht, 1996.

P. Visser. *Knowledge Specification for Multiple Legal Tasks: A Case Study of the Interaction Problem in the Legal Domain*. Kluwer Law Int, 1995.

P. Visser, R. Kralingen, and T. Bench-Capon. A method for the development of legal knowledge systems. In *Proceedings of the Sixth International Conference on Artificial Intelligence and Law*, pages 151–160. ACM Press, 1997.

R. Wieringa. *Requirements Engineering: Frameworks for Understanding*. John Wiley & Sons, Inc. New York, NY, USA, 1996.

P. Wolper. Constructing automata from temporal logic formulas: A tutorial. In *Lectures on Formal Methods and Performance Analysis*, number 2090 in LNCS, pages 261–277. Springer-Verlag, New York, 2002.

M. Wooldridge. Agent-based software engineering. *IEEE Proceedings on Software Engineering*, 144(1):pages 26–37, 1997.

M. Wooldridge and P. Ciancarini. Agent-oriented software engineering. In S. K. Chang, editor, *Handbook of Software Engineering and Knowledge Engineering*, volume 1, pages 507–522. World Scientific Publishing Co., 2002.

M. Wooldridge and N. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 3(3):pages 115–152, 1995.

G. von Wright. Deontic logic. *Mind*, 60:pages 1–15, 1951.

B. Yu and M. Singh. Emergence of agent-based referral networks. In *Proceedings of the first International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1208–1209. Bologne, Italy, 2002.

Index

Summary

The motivation for the research presented in this thesis is the ANITA project, which is a multidisciplinary joint venture of legal and artificial intelligence departments of four universities in the Netherlands (consisting of Rijksuniversiteit Groningen, Universiteit Leiden, Universiteit Maastricht and Universiteit Utrecht). ANITA stands for Administrative Normative Information Transaction Agents, and aims to provide a solution, based on multiagent technology, for the problem that exists in information exchange between the different police districts in the Netherlands. In this domain the main challenges concern both the shortage of information (not being able to find legally relevant data that should be available) as well as abundance of information (for example violating privacy rights). These issues are of great consequence in the domain of police and judicial intelligence. The ANITA framework is based on administrative agents that decide, based on norms, whether to allow transactions of information.

In the ANITA scenario, where agents autonomously decide whether or not to share (privacy sensitive) information based on the applicability of (local) norms, a global frame for the enforcement of (global) norms was needed. Since all agents, though each bound to local procedures and rules, have to adhere to the global regulations as given by law (e.g. the police file act, and privacy laws), it has to be checked (at real-time) whether the information transactions are not in conflict with the global laws that hold for the domain. In most software and agent methodologies, such regulations are seen only as extra requirements in the analysis phase of the system, and are thus hard-coded into the software or the agents themselves. If, however, the regulations change it becomes very hard to track all changes to be done in the implementation, as there is no explicit representation of the norms (that is, since all norms are embedded in the agent's design and code, all the design steps have to be checked and all the code verified to ensure compliance to the new regulations). The alternative is to have an explicit representation

of the norms, but this approach requires some form of enforcement to ensure the compliance instead.

This alternative can be found in the introduction of an electronic institution. An electronic institution, similar to their human counterparts, is an entity defining a set of norms over the behaviour of individuals inside the institution. Electronic institutions provide a safe environment mediating the interaction of agents, where the expected behaviour of the agents is expressed by means of an explicit specification of the norms. The introduction of an electronic institution in a highly-regulated domain such as the ANITA scenario requires us to solve issues related to the abstractness of human regulations, the lack of operational information and the implementation of norm enforcement from an institutional perspective. In this thesis we solve these problems by the introduction of a framework for making the connections between the norms and the agent practice explicit.

An electronic institution built based on a specification of norms taken from human laws and regulations is characterised by the normative specification and its procedures. Both the normative specification and the procedures, expressed in protocols that the agents can use to achieve common tasks, are derived from the laws and regulations that govern the domain. To ensure that none of these laws are broken by any of the agents participating in the institution, some form of enforcement is required. This can either be done by restricting the agents to specific procedures (that are known to be norm-compliant) or by monitoring the behaviour of the agents and punishing them in the case they violate a norm. We argue that the latter holds many benefits, since restricting agents to pre-defined procedures severely reduces the autonomy of the agents participating in the institution (restricting agents to such protocols only allows them to act in the manners defined on forehand, thus making them unable to handle new and unforeseen situations). This does, however, signifies the implementation of an active norm enforcement based on the detection of violations and reactions to violations, which has not been done before, and can be hard because of the abstractness of the norms and the lack of operational information in norms (norms do not explicitly define how they should be implemented). We solve this problem by introducing a formal representation of norms based on a representation in deontic logic which is annotated with fields that contain all the (operational) information necessary to implement the norms from an institutional perspective.

Even though the agents are not restricted to pre-defined procedures, the introduction of protocols for an institution still holds an advantage. Protocols can be used by the agents as a default manner to achieve certain (frequently done) tasks in the institution, without having to take the norms into notion (the norm compliance of the agent's actions should follow from the correct execution of the protocol). Designing protocols for highly-regulated domains is, however, a very hard task because of the gap that exists between the normative specification of the domain (in abstract and vague norms) and the procedural level of the agent practice. To bridge this gap, and allow for the design of protocols for electronic institutions, we introduced a (semi)automatic manner to translate the specifications given by the norms into a concrete pattern that can be used to create a practical protocol. This translation uses an idea based on the approach taken by human institutions, where the gap between the law and the practice is bridged by the introduction of regulations that, while still somewhat abstract, give a concrete meaning to some of the vague and abstract notions and actions expressed in the laws. With the introduction of an intermediate level of landmarks between the norms and the practice, where the landmarks express the important steps that any protocol created should contain, it becomes possible to create protocols based on the (vague and abstract) normative specification of the domain. In case, instead of designing protocols for the institution by the manner described above, protocols are taken from the human practice and translated to the electronic institution, it has to be guaranteed that the protocols do not break any of the norms themselves. That is, following a protocol by the letter should not make the agent violate any of the norms. To ensure that a protocol does not break any of the norms, its norm compliance has to be formally verified.

These elements, the implementation of active norm enforcement and the design of protocols for electronic institutions, form the basis of the framework that links the laws, expressed in a normative specification, to the agent practice. While the framework is not complete, it makes the first steps that are needed and tries to cover the most important aspects of institution design. The creation of this framework, however, gave us insight into the relations that exist between laws and electronic institutions, and helped us design methods for the design and verification of protocols. Furthermore, the implementation of norms, as proposed in this framework, by the creation of a mechanism of active norm enforcement ensures a good balance between autonomy and conformity of the agents participating in the electronic institution.

Samenvatting

De belangrijkste motivatie voor het onderzoek gepresenteerd in dit proefschrift is het ANITA project. Dit project is een multidisciplinaire samenwerking tussen de rechten en kunstmatige intelligentie departementen van vier Nederlandse universiteiten (deelnemend zijn Rijksuniversiteit Gronigen, Universiteit Leiden, Universiteit Maastricht en Universiteit Utrecht). ANITA staat voor Administratieve Normatieve Informatie Transactie Agents, en heeft als doel om een oplossing te bieden, gebaseerd op agent technologie, voor het informatie uitwisselingsprobleem dat bestaat tussen de verschillende politie corpsen in Nederland. De uitdagingen van dit domein zijn voornamelijk zowel het ontbreken van informatie (niet instaat zijn om legaal relevante data te vinden die beschikbaar zou moeten zijn) als een overvloed aan informatie (bijvoorbeeld het schenden van privacy rechten). Deze problemen hebben grote gevolgen in de domeinen van politie- en jurisdische inlichtingen. Het ANITA framework is gebaseerd op administratieve agents die beslissen op basis van normen of transacties van informatie worden toegelaten of niet.

In de ANITA scenario, waar agenten autonoom beslissingen nemen over het wel of niet delen van (privacy gevoelige) informatie op basis van de toepasbaarheid van (lokale) normen, was het noodzakelijk om een globaal kader te maken voor de handhaving van (globale) normen. Aangezien alle agents, hoewel elk gebonden aan lokale procedures en regels, zich moet houden aan globale reguleringen gegeven door de wet (bijvoorbeeld de Wet Politieregisters of privacy wetten), moet er gecontroleerd worden (in real-time) of the informatie transacties niet in conflict zijn met de globale wetten die van toepassing zijn voor het gegeven domein. De meeste software en agent methodologieën zien deze reguleringen enkel as extra elementen van de analyse fase van het systeem, en de reguleringen worden dan ook als deel van de software of agents zelf geprogrammeerd. Maar, als de reguleringen veranderen wordt het bijzonder lastig om alle veranderingen na te gaan die gedaan moeten worden in de implementatie, aangezien er geen expliciete representatie van de normen is (omdat alle normen in de code en het ontwerp van de agents is geïntegreerd, moeten alle ontwerpstappen bekeken worden en alle code worden gecheckt om overeenkomst met de nieuwe reguleringen te kunnen garanderen). Het alternatief is om een expliciete representatie te maken van de normen, maar deze aanpak vereist een of andere vorm van norm handhaving om te garanderen dat aan de normen wordt voldaan.

Dit alternatief is te vinden in de introductie van een elektronische institutie. Elektronische instituties, overeenkomstig met hun menselijke soortgelijken, is een entiteit dat een verzameling normen definieert dat het gedrag van de individuen in de institutie beschrijft. Elektronische instituties creeëren een veilige omgeving voor agents om in te werken, en beschrijven het gewenste gedrag van de agents door middel van een expliciete specificatie van de normen. De introductie van een elektronische institutie in een zeer regulatief domein zoals die van het ANITA scenario, vereist dat wij problemen oplossen zoals de abstractheid van wetten, het ontbreken van operationele informatie en de implementatie van norm handhaving vanuit het perspectief van de institutie zelf. In dit proefschrift hebben we geprobeerd deze problemen op te lossen door het invoeren van een framework voor het expliciet maken van de verbindingen tussen de normen en de (agent) praktijk.

Een elektronische institutie die gebouwd is op basis van een specificatie van normen uit wetteksten en reguleringen wordt gekarakteriseerd door deze normatieve specificatie en zijn gebruiken. Zowel de normatieve specificatie als de gebruiken, uitgedrukt in protocollen die door de agents worden gebruikt om veel voorkomende taken te voltooien, worden afgeleid uit de wetten en reguleringen die gelden voor het domein. Om ervoor te zorgen dat geen van de wetten worden overtreden door een van de agents in het systeem is een of andere vorm van handhaving noodzakelijk. Deze handhaving kan ofwel gebeuren door het beperken van de agents tot specifieke procedures (waarvan bekend is dat ze geen wetten overtreden) of door het controleren van het gedrag van de agents en het toedienen van straffen in het geval dat een norm wordt overtreden. De laatste van deze twee mogelijkheden heeft vele voordelen, ondermeer omdat het beperken van de agents tot vooraf gedefinieerde procedures een grote beperking oplegt op de autonomie van de agents (het beperken van de agents tot dergelijke protocollen stelt hen enkel in staat om te handelen in die manieren die vooraf zijn bedacht, en maakt hen dus onbekwaam in nieuwe en onvoorziene situaties). Dit benadrukt de noodzaak van een implementatie van een actieve norm handhaving gebaseerd op het detecteren van en het reageren op overtredingen, wat voorheen nog niet gedaan was, en wat bijzonder lastig kan zijn wegens de abstractheid van de normen en het ontbreken van operationele informatie in de normen (normen geven geen expliciete definitie over hoe de norm zou moeten worden geïmplementeerd). Wij lossen dit probleem op door de invoering van een formele representatie van normen gebaseerd op een representatie in deontische logica welke wordt aangevuld met extra velden die de (operationele) informatie bevatten welke nodig is voor de implementatie van de normen vanuit een institutioneel perspectief.

Hoewel we de agents niet willen beperken tot vooraf gedefinieerde procedures is de introductie van protocollen in een institutie nog steeds zinvol. Protocollen kunnen namelijk door de agents worden gebruikt als standaard manier om bepaalde (veel voorkomende) taken te volbrengen in de institutie, zonder daarbij zichzelf te veel druk te maken over de normen (het voldoen aan de normen zou moeten volgen uit de correcte uitvoering van het protocol). Het ontwerpen van protocol voor zeer regulatieve domeinen is echter een moeilijke taak wegens de kloof die er bestaat tussen de normatieve specificatie van het domein (in abstracte en vage normen) en het procedurele niveau van de agent praktijk. Om deze kloof te overbruggen, en om dus protocollen te kunnen ontwerpen voor elektronische instituties, introduceren wij een (semi-)automatische methode om de normatieve specificatie te vertalen naar een concreet patroon dat gebruikt kan worden om een praktisch protocol te maken. Deze vertaling gebruikt een idee dat afgeleid is van de manier waarop de kloof tussen normen en praktijk wordt opgelost in de normale wereld, waar de kloof wordt overbrugd door de introductie van reguleringen, die, hoewel nog steeds enigszins abstract, concrete informatie geven over sommige vage en abstracte concepten en acties in de wet. Door het maken van een tussenlaag van landmarks tussen de normen en de praktijk, welke uitdrukt de belangrijke stappen die elk protocol zou moeten bevatten, wordt het mogelijk om protocollen te maken gebaseerd op de (vage en abstracte) normatieve specificatie van het domein. Als echter in plaats van het ontwerpen van nieuwe protocollen ervoor wordt gekozen om protocol over te nemen van de werkelijke wereld, zal het moeten worden vastgesteld dat deze protocollen geen van de normen overtreedt. Dit betekent dat wanneer het protocol tot op de letter nauwkeurig wordt uitgevoerd er geen overtredingen mogen plaatsvinden. Om dit te kunnen garanderen zal het protocol formeel moeten worden gecheckt.

Deze elementen, de implementatie van een actieve norm handhaving en het ontwerpen van protocol voor elektronische instituties, vormen de basis van het framework dat de wetten, uitgedrukt in een normatieve specificatie, verbind aan de agent praktijk. Hoewel dit framework nog niet volledig is, geeft het de eerste stappen die benodigd zijn and probeert de meest belangrijke aspecten van institutie ontwerp te omvatten. Het maken van dit framework gaf ons echter vele inzichten in de relaties die bestaan tussen wetten en elektronische instituties, en hielp ons bij het bedenken van methoden voor het ontwerpen en verifiëren van protocollen. Bovendien garandeert de implementatie van normen, zoals voorgesteld in dit framework door middel van het creeëren van een mechanisme van actieve norm handhaving, een goede balans tussen de autonomie en de conformiteit van de agenten die deelnemen in de elektronische institutie.

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, 2003-1

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 e-cology 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 Jovanoviç, *To Who It May Concern - Addressee Identification in Face-to-Face Meetings*, UT, 2007-08

Mark Hoogendoom, *Modeling of Change in Multi-Agent Organizations*, VU, 2007-09

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