Sequential Decision Making in Normative Environments

Universidad
Rey Juan Carlos

Sequential Decision Making in Normative Environments

Tesis presentada por

Moser Silva Fagundes

Departamento de Arquitectura y Tecnología de Computadores,
Ciencias de la Computación e Inteligencia Artificial

para la obtención del título de
Doctor en Informática

D. Sascha Ossowski, Professor Catedrático de la Universidad Rey Juan Carlos, con NIE X-1805919-M,

CERTIFICA: Que D. Moser Silva Fagundes, Ingeniero en Informática, ha realizado en el Dpto. de Arquitectura de Computadores y Ciencias de la Comunicación e Inteligencia Artificial, bajo su dirección, el trabajo correspondiente a la tesis doctoral titulada:

Sequential Decision Making in Normative Environments

Revisado el presente trabajo, estima que puede ser presentado al tribunal que ha de juzgarlo. Y para que conste a efecto de lo establecido en la normativa reguladora del tercer ciclo de la Universidad Rey Juan Carlos, autoriza su presentación.

Móstoles, 11 de Octubre de 2012.

Fdo.: Sascha Ossowski

Dedicated to my loving parents.

Acknowledgments

Writing a thesis is often described as a solitary and lonely endeavor, and at times it certainly was, but I never would have completed it without the support and assistance of many, many people.

First I would like to thank my advisor, Professor Sascha Ossowski, for his great support and guidance. Despite his overfilled schedule, he has always found the time to help me throughout the course of this research, coming up with fruitful ideas and enthusiasm. Professor Holger Billhardt deserves sincere appreciation for his help during the early stages of my doctorate research. I am grateful to my colleagues at URJC – Alberto, Carmen, Carlos, José Javier, José Santiago, Levan, Luis, Marin, Matteo, Radu, Ramón, Roberto, Rubén and Zijie. I am thankful to Professor Michael Luck and Professor Simon Miles, who supervised my research stay in London, and Professor Jesús Cerquides and Professor Pablo Noriega, who received me in Barcelona and helped me with useful suggestions for my research. I would to express my gratitude to Professor Rosa Vicari and Professor Ricardo Silveira for supporting my past research and teaching my first lessons on multiagent systems.

I am grateful to my family – Silvio (*in memoriam*), Epitácio (*in memoriam*), Odila, Neli, Lucimar, Dieisson, Eva, Iuri, Hélio and Obegair – for their continuous support and encouragement over the last years. I am also grateful to my friends, specially Alessandro, Núria, Mário, Vinícius, Rafael and Pilar, who have been supportive in the last years, helping me to disconnect from the thesis when I needed fresh air.

I want to thank Debora, *minha prenda*. It would take another thesis to recount all of the ways she has lovely supported me. I am lucky to have you in my life and I look forward to more adventures together when I go back home.

Last but not least, I would like to thank my parents, José and Carmiranda, for their belief in me and for always being there for me. You made this possible.

Thesis advisor
Sascha Ossowski

Author
Moser Silva Fagundes

Abstract

Normative multiagent systems are a vibrant field of research that has received much attention in recent years. In particular, a broad variety of norm-aware agent models and architectures have been developed, aimed at implementing the normative reasoning of agents with different levels of autonomy and in different types of environments. However, approaches that allow autonomous agents to generate complex plans in dynamic and non-deterministic normative environments are rare, as they are notoriously difficult to set-up and hard to evaluate in a quantitative manner.

This thesis introduces the *Normative Markov Decision Processes* (NMDPs), an extension of the *Markov Decision Processes* (MDPs) for modelling norm-aware rational agents acting in normative stochastic environments, as well as two utilitarian models of normative reasoning, pertaining to *self-interested* and *norm-compliant* agents. While the self-interested agents prioritize the maximization of utilities over the compliance with norms, the norm-compliant agents prioritize the norm-abiding behaviour over the utility maximization.

Combining MDPs with normative agent models revealed a significant synergistic potential. On the one hand, norms help shaping the behaviour of rational normative agents with an MDP-based world model, fostering coordination in a multiagent setting, and achieving computational leverage by pruning the search space for the agents' policy construction. On the other hand, MDPs are a principled way for norm-aware agents to model the uncertainty in their environment, and to provide effective general algorithms to determine rational action plans in such a setting. This, in turn, makes it possible to perform quantitative analyses both at agent and at system level.

To validate the approach, several experiments were performed in a simulated motion environment, measuring the performance of different populations of agents in relation to specific controlled settings. Furthermore, by means of a case study in the domain of aerospace aftermarkets, the capability of the NMDP approach to model relevant properties of a real-world scenario and to reason about contracts within such a setting has been demonstrated.

Director
Sascha Ossowski

Autor
Moser Silva Fagundes

Resumen

Los sistemas multiagentes normativos son un vibrante campo de investigación que ha recibido bastante atención en los recientes años. En particular, una amplia variedad de modelos normativos de agente fueron desarrollados con la intención de implementar razonamientos normativos en agentes con diferentes niveles de autonomía en diferentes tipos de entorno. Todavía, propuestas que permitan que agentes autónomos generen planos complejos en entornos dinámicos y no deterministas son raras, una vez que ellos son notablemente difíciles de estructurar y evaluar en términos cuantitativos.

La presente tesis introduce los *Normative Markov Decision Processes* (NMDPs), una extensión de los *Markov Decision Processes* (MDPs) para modelar agentes racionales normativos operando en entornos estocásticos regulados por normas, bien como dos modelos de raciocinio normativo utilitario, perteneciendo a agentes egoístas (*self-interested*) y agentes que siempre cumplen las normas (*norm-compliant*). Mientras los agentes *self-interested* priorizan el incremento de la utilidad, los agentes *norm-compliant* priorizan el comportamiento normativo.

La combinación de MDPs con modelos normativos de agente ha revelado un significante potencial sinérgico entre esas dos áreas de investigación. Por un lado, las normas nos permiten moldear el comportamiento de los agentes racionales basados en MDPs, impulsar la cooperación en un ámbito multiagente, y acotar el espacio de búsqueda en la construcción de los planes con la intención de reducir el tiempo necesario para computar un plan óptimo. Por otro lado, los MDPs facilitan la representación de conocimiento incierto y el desarrollo de algoritmos generales efectivos para determinar planos de acción en entornos no deterministas. Eso, en contrapartida, hace posible las evaluaciones cuantitativas, tanto de los agentes como del sistema.

Para validar nuestra abordaje, hemos realizado varios experimentos en un entorno de movilidad simulado, en el cual hemos medido el desempeño de diferentes poblaciones de agentes en relación a determinados parámetros controlados. Además, por medio de un estudio de caso en un dominio de mercados secundarios aeroespaciales, hemos demostrado la aptitud de los NMDPs para modelar propiedades relevantes de un escenario del mundo real y razonar sobre contratos.

Contents

List of Figures

List of Tables

Part I

Concepts and theories

Chapter 1

Introduction

Por mais longa que seja a caminhada, o mais importante é dar o primeiro passo.

Vinícius de Moraes

1.1 Overview

Everyday, we continuously face sequential decision making problems. For example, when we drive from one place to another, we have to decide which route to take; when a corporation intends to launch a new product in the market, it decides from which suppliers to purchase the components needed to manufacture that new product; when a student engages in a doctorate program, he has to decide his research line.

The essence of sequential decisions is that our choices now depend on the decisions we made in the past, and the outcome of decisions made now can affect our decisions to be made in the future. That is, our best decision now depends on future situations and how we face them. In computer science, the problem of developing mechanisms by which agents can make sequential decisions has been extensively investigated by means of Markov Decision Processes (MDPs) [6, 61, 95, 10, 93], which have demonstrated to be well-suited for quantitative analysis of agent performances, and powerful enough to express uncertainty. In principle, MDPs can be utilized to generate and evaluate sequential decisions for agents, however, the problem of computing optimal sequential decisions is computationally expensive [90, 75].

In some situations, norms play a decisive role in sequential decision making. They influence our choices by providing rules that restrain our options to achieve some kind of coordination with others. For example, when we drive from one place to another, there are norms regulating the traffic in the streets; when corporations purchase products from each other, the interactions between them are most likely to be regulated by contracts; when students engage in doctorate programs, they are expected to comply with the university norms.

In the above examples, the participants are assumed to be autonomous to decide whether or not to comply with the norms. To make such decisions in simple environments may not be a problem. However, the situation changes drastically in complex environments where uncertainty is present. In this case, they will not always be sure about the outcomes, rewards and costs of their actions, and they will not always be sure when norm violations will be detected and punished.

In the past decades, the artificial intelligence community proposed several models for enabling normative reasoning in intelligent agents [28, 42, 24, 68, 2, 123]. Arguably, the main advantage of these normative agent models come from the fact that we can employ them to design multiagent systems where the interactions between the agents can be coordinated by means of norms that restrict their activities [107, 3].

Studying the state of the art in MDPs and normative agent models we have noticed a lack of work on the topic of combining both approaches to decision making. In order to bridge this gap, this thesis focuses on a class of problems in which utilitarian agents, modeled with Markov Decision Processes, make sequential decisions in environments regulated by norms.

We believe that by bridging this gap between MDPs and normative agent models, each research field could benefit from the native strengths of the other. Norms could help with coming up with a way to shape the agents' behavior, enabling coordination and improving the tractability of MDPs by restricting the state space to be explored. On the other side, MDPs can help with their ability to perform quantitative evaluations, which in this case, could be used to evaluate the impact of norms on the agents' utilities.

1.2 Research objectives

The main research objective of this thesis is to develop and evaluate computational models of norm-aware rational agents capable of generating complex plan of action in dynamic and non-deterministic normative environments

This main objective can be divided into the following specific objectives:

(i) To formalize a framework that allows for modeling norm-aware utilitarian agents in stochastic environments where norms regulate the agents' activities.

(ii) To develop models of normative reasoning under uncertainty that enable norm-aware utilitarian agents to incorporate norms into their knowledge and decide whether or not to comply with the norms regulating the stochastic environment they are in.

(iii) To investigate the impact of the proposed models of normative reasoning on the sequential decisions and utilities of norm-aware utilitarian agents.

(iv) From a global perspective, to study how norms can be employed to coordinate norm-aware utilitarian agents.

1.3 Methodological remarks

To achieve objective (i), we generalize an existing framework, namely Markov Decision Processes, by including normative structures and detection probabilities of norm violations. The formalization of this new framework has been made by means of set theoretical notions, and the notion of norms has been borrowed from previous work on normative systems.

To develop the models of normative reasoning proposed in objective (ii), we have relied on the framework mentioned in objective (i). The structure of the framework allowed us to develop general algorithms implementing different notions of rationality. Once the normative reasoning has been encoded in the form of algorithms, we built them into particular agent models, namely self-interested and norm-compliant.

The objectives (iii) and (iv) have been achieved through experiments performed in simulated normative multiagent systems with controlled experimental settings and well-defined performance criteria.

1.4 Roadmap

This section provides an overview of the structure of this thesis, which is divided in four parts detailed as follows:

- Part I – *Concepts and theories.*
  The first part contains this introductory chapter and Chapter 2, which describes the background that underlies this thesis with an emphasis on Markov Decision Processes, Bayesian networks, normative multiagent systems and norm-aware agent models, ending with an analysis of the current approaches for normative reasoning in autonomous agents.

- Part II – *NMDP framework and agent models.*
  The second part introduces our proposal for modeling normative rational agents. Chapter 3 addresses objective (i) by formalizing the *Normative Markov Decision Processes* (NMDPs), a framework for modeling norm-aware utilitarian agents. In Chapter 4, we address objective (ii) by developing two models of normative reasoning that use the NMDP framework for sequential decision making.

- Part III – *Experimentation.*
  Together with the second part, the third part constitutes the core parts of this thesis. In this part we address objectives (iii) and (iv). Chapter 5 presents and discusses a series of experiments performed in a simulated motion environment. Chapter 6 demonstrates, by way of a case study in a simulated aerospace aftermarket, the applicability of the NMDP framework to model and reason about contracts in stochastic environments.

- Part IV – *Conclusion.*
  In the last part, Chapter 7 draws the main conclusions of this thesis, as well as some limitations of our approach and future research directions that we intend to look at.

Chapter 2

Background

There are no shortcuts in evolution.

Louis Dembitz Brandeis

This chapter presents a short description of the background that underlies this thesis. The research areas composing this background and their relationship are shown in Figure 2.1. Section 2.1 concerns with sequential decision problems modelled as *Markov Decision Processes*, which in the context of this thesis, are seen as a formalism to create autonomous agents. Section 2.2 presents the *Bayesian networks*, utilized in compact representations of Markov Decision Processes. Section 2.3 briefly presents the usage of *norms* in *multiagent systems*, while Section 2.4 makes a survey on *norm-aware agent models*. Section 2.5 analyzes and compares these norm-aware agent models. Finally, Section 2.6 promotes a discussion, highlighting the *gap* between the norm-aware agent models developed so far and the generalizations and computational methods for Markov Decision Processes.

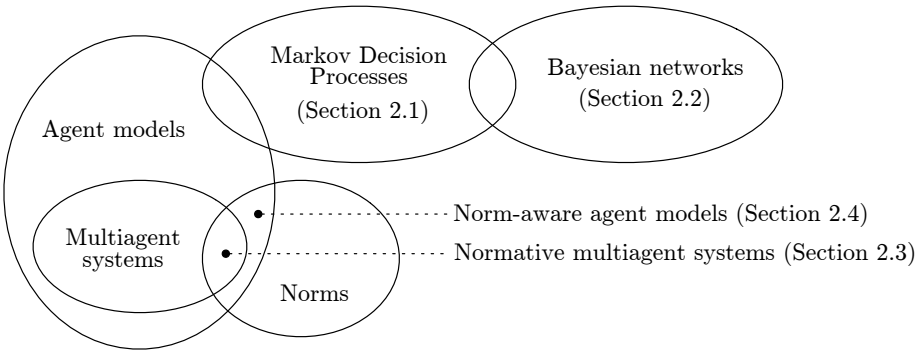


Figure 2.1: Research areas composing the background that underlies this thesis.

2.1 Markov Decision Processes

Markov Decision Processes (MDPs for short), named after Andrey Markov[1], provide a formal mathematical framework for modeling sequential decision making in stochastic environments. The framework itself is fairly simple and it is only when we attempt to use these processes to determine optimal behavior that any complications appear.

The formal model of MDP used in this thesis, given in Definition 1, has a *finite* number of states and actions. The MDP framework includes a capability function that indicates which actions are admissible in every state. The *Markov property* holds, that is, the state-transition probabilities from any given state depend only upon the state and not upon the previous history. In this case, there is no need for retaining any information about the history of its past actions in order to make optimal decisions. Finally, the reward function indicates the immediate reward of executing an action in a given state and then making a state-transition. Some definitions of reward allow the reward to depend only on the current state and/or action, which does not change the problem in any fundamental way.

[1]Andrey Andreyevich Markov (14 June 1856 – 20 July 1922) was a Russian mathematician, particularly remembered for his study of Markov chains, sequences of random variables in which the future variable is determined by the present variable but is independent of the way in which the present state arose from its predecessors. This work founded a completely new branch of probability theory and launched the theory of stochastic processes.

Definition 1 (**Markov Decision Process**). *A Markov Decision Process (MDP) is defined as a tuple* $\langle \mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{T}, \mathcal{R} \rangle$ *where:* $\mathcal{S}$ *denotes a finite set of states of the world;* $\mathcal{A}$ *denotes a finite set of actions;* $\mathcal{C} : \mathcal{S} \to \mathcal{A}$ *is a capability function that denotes the set of admissible actions in a given state* ($\mathcal{C}(s)$ *corresponds to the set of admissible actions in the state* s*);* $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ *is a state-transition function* ($\mathcal{T}(s, a, s')$ *indicates the probability of executing* a *at* s *and ending at* s'*); and* $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ *is a reward function that determines the expected immediate reward* ($\mathcal{R}(s, a, s')$ *corresponds to the one gained by the agent for executing* a *at* s *and ending at the state* s'*).*

MDPs have been extensively studied and used in several fields, such as automated control [10, 93], economics [4] and artificial intelligence [102]. Here, the MDPs are built in *autonomous agents*, which use this formalism to represent their knowledge about the environment they are living in.

2.1.1 Optimality in sequential decision making

Before we define optimality in sequential decision making with the MDP framework, consider the following assumptions about the process:

- The agent always knows *exactly* what state of the environment it is in. Thus, each decision is taken with exact knowledge of the current system state.
- The process is *discrete-time* and *stochastic*. At each time step, the process is in some state, and the decision maker may choose any action that is available in this state. The result of the chosen action in the given state is determined by the state-transition function.
- The total reward maximization is a *discounted* problem, where γ, refereed to as *discount factor*, is a positive scalar between 0 and 1.
- There is an *infinite-horizon* for decision making. Note that it does not necessarily mean that all state sequences are infinite, it just means that there is no fixed deadline. In particular, there can be finite state sequences in an infinite-horizon MDP containing an absorbing state.

In finite state, completely observable, discrete-time, discounted and infinite-horizon MDPs, the solution has the form of a stationary policy as formalized in Definition 2.

Definition 2 (**Stationary policy**). *A stationary policy for an MDP is defined as a function* $\pi : \mathcal{S} \to \mathcal{A}$*, and* $\pi(s)$ *corresponds to the action to be executed in the state* s*.*

If we were simply interested in the immediate rewards, the problem would have a trivial solution of always selecting the action with the highest immediate reward:

$$\pi(s) = \underset{a \in \mathcal{C}(s)}{\operatorname{argmax}} \, \mathcal{IR}(s, a),$$

where $\mathcal{IR}(s, a)$ is the immediate reward of the action a in the state s:

$$\mathcal{IR}(s, a) = \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s') \mathcal{R}(s, a, s')$$

The problem of maximizing the total rewards in an *infinite-horizon* and *discounted* setting is, however, more complex due to the trade-off between the immediate rewards and the rewards that are received in the future. Among the many ways for making the trade-off between immediate and future rewards[2], the work developed in this thesis employs the *expected future discounted reward* with respect to the *indefinite* sum

$$E\left[\sum_{t=0}^{\infty} \gamma^t \ \mathcal{IR}(s_t, a_t)\right],$$

where t denotes the time step, γ is the discount factor, s_t and a_t are the current state and the action to be executed in the time step t, respectively, and teh function $\mathcal{IR}(a_t, s_t)$ gives the immediate reward of a_t in s_t. Note that we impose the constraint $0 \leq \gamma < 1$ to guarantee that the expectation is bounded. On this basis we can define an *optimal policy* as

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \ E\left[\sum_{t=0}^{\infty} \gamma^t \ \mathcal{IR}(s_t, \pi(s_t))\right]$$

The next section describe algorithms for computing optimal policies.

2.1.2 Computational methods to compute optimal policies

The work carried out by Bellman [6] is of great significance for the research on sequential decision-making problems, as he proposed the dynamic programming approach in general and the Value Iteration algorithm in particular. The fundamental idea of the Value Iteration consists of computing the utility of each state, and then, utilizing these utilities to create an optimal policy. The utility of a state is calculated in terms of the utility of the state sequences that might follow it, which depends on the policy that is executed. So the utility of a state with respect to a policy π is:

[2]Other optimality criteria not explored in this thesis, such as *average reward* obtained per time step and the *expected future discounted reward* with respect to the finite horizons, are detailed in [10, 93].

$$\mathcal{U}^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t \ \mathcal{IR}(s_t, \pi(s_t)) \mid s_0 = s\right],$$

If the agent executes an optimal policy π^*, the utility of a state is $\mathcal{U}^{\pi^*}(s)$, hereafter written simply as $\mathcal{U}(s)$. The action to be executed in each state is selected by means of the maximum expected utility principle, that is:

$$\pi^*(s) = \underset{a \in \mathcal{C}(s)}{\operatorname{argmax}} \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s')\mathcal{U}(s')$$

Given that the utility of a state is the expected sum of discounted rewards from that state onwards, then we can say that there is a relationship between the utility of a state and the utility of its neighbors. This relationship is formalized in the Bellman equation, named after its discoverer:

$$\mathcal{U}(s) = \max_{a \in \mathcal{C}(s)} \mathcal{IR}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s')\mathcal{U}(s')$$

The Bellman equation provides the basis for the Value Iteration, specified in Algorithm 1, an iterative algorithm that computes the *expected utility* of each state using the expected utilities of the neighboring states. Initially, the utility functions $\mathcal{U}$ and $\mathcal{U}'$ return zero for all states. For every state, the algorithm updates the utility of the states (L5), and then, it computes the maximum change υ in the utility of any state in the current iteration (L6–L7). The algorithm iterates until the utilities calculated in two successive iterations are close enough ($\upsilon < \epsilon$, where ϵ is the maximum error).

Algorithm 1 – Value Iteration

Input: $\langle \mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{T}, \mathcal{R} \rangle$ (MDP) and ϵ (Maximum error).
Local variables:
$\mathcal{U}$ (Utility function, initially zero for all states),
$\mathcal{U}'$ (Temporary utility function, initially zero for all states),
γ (Discount factor) and
υ (Maximum change in the utility of any state in an interation).

1. **repeat**
2. $\quad \mathcal{U} \leftarrow \mathcal{U}'$
3. $\quad \upsilon \leftarrow 0.0$
4. $\quad$ **for all** $s \in \mathcal{S}$ **do**
5. $\quad\quad \mathcal{U}'(s) \leftarrow \max_{a \in \mathcal{C}(s)} \mathcal{IR}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s')\mathcal{U}(s')$
6. $\quad\quad$ **if** $|\mathcal{U}'(s) - \mathcal{U}(s)| > \upsilon$ **then**
7. $\quad\quad\quad \upsilon \leftarrow |\mathcal{U}'(s) - \mathcal{U}(s)|$
8. **until** $\upsilon < \epsilon$
9. **return** $\mathcal{U}$

In infinite horizon discounted problems, this algorithm is guaranteed to converge to an optimal utility function and to an optimal stationary policy in a finite number of value iterations. The rates of convergence, stopping criteria, and many other results relating to the convergence behaviour of this algorithm can be found in [94].

While the Value Iteration algorithm explores the *utility space*, the Policy iteration [61] does it in the *policy space*. The fundamental idea of the Policy Iteration, shown in Algorithm 2, consists of generating a sequence of stationary policies, each with improved expected utility over the preceding one. The algorithm starts by evaluating the current policy π by means of the *policy-evaluation* routine (L3). There are several approaches to implement this routine, such as solving the linear system of simplified Bellman equations and interactively calculating the utilities by means of value iterations. Then, in the policy improvement stage (L5–L8), the algorithm computes an improved policy using one-step look ahead based on the utility function $\mathcal{U}$ resulting from the policy evaluation. The algorithm ends when the policy stabilizes.

Algorithm 2 – Policy Iteration

Input: $\langle \mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{T}, \mathcal{R} \rangle$ (MDP) and π' (Initial policy).
Local variables:
$\mathcal{U}$ (Utility function, initially zero for all states),
$\mathcal{U}'$ (Temporary utility function, initially zero for all states) and
π (Policy).

1. $\pi \leftarrow \pi'$
2. **repeat**
3. $\quad \mathcal{U} \leftarrow \mathit{policy\text{-}evaluation}(\pi, \mathcal{U}, \mathrm{MDP})$
4. $\quad \mathit{unchanged} \leftarrow \mathit{true}$
5. $\quad$ **for all** $s \in \mathcal{S}$ **do**
6. $\quad\quad$ **if** $\max_{a \in \mathcal{C}(s)} \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s')\mathcal{U}(s') > \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \pi(s), s')\mathcal{U}(s')$ **then**
7. $\quad\quad\quad \pi(s) \leftarrow \operatorname{argmax}_{a \in \mathcal{C}(s)} \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s')\mathcal{U}(s')$
8. $\quad\quad\quad \mathit{unchanged} \leftarrow \mathit{false}$
9. **until** *unchanged*
10. **return** π

Puterman and Shin [95] studied a class of Policy Iteration algorithm, referred to as Modified Policy Iteration, where the *policy evaluation* is performed by some number of value iterations using the simplified Bellman update[3]:

[3]This update is called *simplified* because the utilities are calculated for a fixed policy instead of computing the maximum of the utilities for admissible actions in each state.

$$\mathcal{U}(s) \leftarrow \mathcal{IR}(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \pi(s), s')\mathcal{U}(s')$$

The number of value iterations performed per policy evaluation determines the *order* of the Modified Policy Iteration algorithm.

2.1.3 Computation leverage techniques

The main problem with MDPs is that, in many real-life domains, the state space is quite large. In these cases, computing an optimal policy can be prohibitively time-consuming. Papadimitriou and Tsilikis [90] analyzed the computational complexity of MDPs, and proved that any MDP can be represented as a linear program and solved in polynomial time for both finite and infinite horizons. However, the order of the polynomials is large enough that theoretically efficient algorithms, such as the ones shown in 2.1.2, are not efficient in practice with large state spaces. For details on the complexity of solving MDPs, see [90, 75].

Out of many approaches for achieving computational leverage with Markov Decision Processes, here we are particularly interested on specialized compact representations, and algorithms employing these representations, that exploit structural properties of the state space. In practice, these compact representations are very convenient as they allow the description of sets of states on the basis of certain properties or features, avoiding this way an exhaustive enumeration. These compact representations, widely known in the AI literature as Factored Markov Decision Processes [20], employ *factored state spaces* and *Dynamic Bayesian networks* [37, 35] to compactly represent state-transition models:

(i) A factored state space is described by way of a set of features $\mathcal{F}$, where each feature $f_i \in \mathcal{F}$ takes on a finite number of values in $\mathcal{V}_{f_i}$ (the domain of the feature f_i). A state is any possible assignment of values to these features, and the state space $\mathcal{S}$ is the cross product of the value spaces for the features:

$$\mathcal{S} = \times_{i=1}^{F} \mathcal{V}_{f_i}.$$

(ii) The Dynamic Bayesian networks [37, 35] are a particular type of Bayesian network (Section 2.2 introduces this knowledge representation) used to represent dependence between features of the state space before and after the occurrence of actions.

In AI efforts for achieving computational leverage in the policy construction with factored state spaces, the emphasis has generally been placed on a particular form of aggregation named *abstraction*. This technique exploits factored representations and allows the grouping of states that are indistinguishable according to certain characteristics, abstracting away some features of the problem deemed irrelevant. A set of states grouped in this manner constitutes a *partition* of the state space and is refereed to as *abstract state*. The underlying idea is that by clustering states, each abstract state can be treated as a single state, alleviating this way the need for computing the value or action of each state individually. Examples of this technique include the structured dynamic programming [21] and the model minimization [36].

Another approach consists of *decomposing* the factored MDPs in various pieces for solving them individually [38]. Here, the underlying idea is to combine the local solutions of each piece in order to generate a global policy. The decompositions are performed on the basis of reachability analyses [18] which determine the decomposition types that can be used (i.e. serial or parallel).

Research on *abstraction* and *decomposition* approaches is vast. Further details and references about these and other computational methods for achieving computational leverage in factored MDPs can be found in the survey by Boutilier et al. [19].

2.1.4 Generalizations of the MDP framework

The MDP framework has been extended in the past decades to serve several purposes. One of the most investigated generalizations of the MDP framework is the *Partially Observable Markov Decision Process* (POMDP) [65], which encodes a decision process in which the agent cannot directly observe the underlying state of the world. Instead, it maintains a probability distribution over the set of possible states using observations and observation probabilities.

Following the progress with applications of MDPs to overwhelm problems involving single agents, many efforts have been made in order to control *collaborative multiagent systems*. Bernstein et al. [9] put forward the *Decentralized Markov Decision Processes* (DEC–MDP) and the *Decentralized Partially Observable Markov Decision Processes* (DEC–POMDP), which generalize MDPs and POMDPs to allow for coordination of distributed collaborative agents. In this research line, several papers propose some improvement of these decentralized models, such as the DEC–MDP with centralized coordination mechanisms [59], the transition independent DEC–MDP [5], the DEC–

POMDP with communication [54] and the *Communicative Multiagent Team Decision Problem* (COM–MTDP) [83].

Still in a collaborative setting, the *Multiagent Markov Decision Processes* (MMDP) [17] model fully cooperative multiagent systems by replacing single actions by joint actions distributed among multiple agents. For computing optimal policies with the MMDP framework, the authors develop a type of value iteration in which the state space of the MMDP is augment with the state of the coordination mechanism adopted. Guestrin et al. [57] develop a principled planning algorithm for cooperative multiagent systems viewed as a single large factored MDP with joint actions. In this approach, the agents select their actions in a cooperative way, using a message passing schema, and complying with some coordination requirements. The key idea of this work is to constrain the set of eligible joint actions in order to avoid the exponential blowup in the action space. In a rather different approach, Bererton et al. [7] utilize an auction mechanism to coordinate multiple MDP agents.

As we can see, there is a significant volume of research on controlling cooperative agents. On the other side, the problem of controlling non-collaborative self-interested agents modeled as MDPs has received less attention. Approaching the problem from the system's perspective, Cavallo et al. [31, 30] develop a centralized incentive mechanism for implementing optimal policies in environments populated by self-interested agents modeled as MDPs. This mechanism has knowledge of each agent's state space and has autonomy to enforce global decision policies, that is, to take decisions on the agents' behalf. However, the mechanism has no access to the agents' current state, which is crucial for identifying imminent undesired states and determining which action should be performed by each agent. To approach this problem, the mechanism requests claims from the agents about their current state, proposes and enforces optimal joint actions based on the agents' claims, and transfer rewards to the agents as an incentive to provide trustful information. Thus the problem of coordinating a group of self-interested agents consists of providing the appropriate incentives so that agents will choose to make truthful reports of local private information.

As a matter of fact, Cavallo et al. [31, 30] do not generalize the MDP framework. Instead, they create an independent agent component for reasoning about the strategy to be used by the agent when informing its current state. This component selects the strategy on the basis of agent's history – past actions, state trajectory and incentives received along this trajectory.

2.2 Bayesian networks

A Bayesian network [92, 63] is a knowledge representation based on probability theory [51] and graph theory [8]. A Bayesian network is a graphical model represented as a direct acyclic graph, where each node in the graph represent a *variable* or *feature*. Each variable has a set of mutually exclusive values, referred to as domain of the variable, and a probability distribution denoting the probabilities that the variable takes on each of its different values. The causal relations between variables are represented as *directed arcs*, leading from the cause variable to the effect variable.

There are two types of variables in a Bayesian network:

- *Unconditioned* variables are represented as prior nodes, which have no parent. Each unconditioned variable has a unconditional probability distribution determining the probability of each value of the variable.

- *Conditioned* variables are represented as non-prior nodes, which have at least one parent node. A conditioned variable has a conditional probability distribution that determines the probability of each value of the variable on the basis of the parents' value.

Formally, a Bayesian network $(\mathcal{W}, \mathcal{P})$ consists of a directed acyclic graph $\mathcal{W}=(\mathcal{F}, \mathcal{L})$ and a set of conditional probability distributions $\mathcal{P}$. In the graph $\mathcal{W}$, the set $\mathcal{F}$ contains the variables (nodes) of the network. The set $\mathcal{L}$ contains the directed arcs in the form (f_i, f_j) denoting the causal relation between $f_i \in \mathcal{F}$ and $f_j \in \mathcal{F}$. Each variable $f_i \in \mathcal{F}$ has a finite set of mutually exclusive values $\mathcal{V}_{f_i}$ and a probability distribution $p_i \in \mathcal{P}$ expressed as a table.

Figure 2.2 illustrates an abstract example of Bayesian network with boolean variables. Looking at the graph, we can see that the variable f_3 is conditioned by f_1, the variable f_4 is conditioned by f_3, and f_5 is conditioned by f_1 and f_2. The variables f_1 and f_2 are unconditional as their value does not depend on the other variables. In a conditioned variable $f_i \in \mathcal{F}$, its table p_i lists the probability that f_i takes on each of its values for each combination of values of its parents. In Figure 2.2, each row of the table of f_3, f_4 and f_5 contains the conditional probability of each variable value for each conditioning case. For instance, given f_1=T, the probability of f_3=T is 0.90; formally, $\mathsf{Pro}(f_3\text{=T} \mid f_1\text{=T}) = 0.90$.

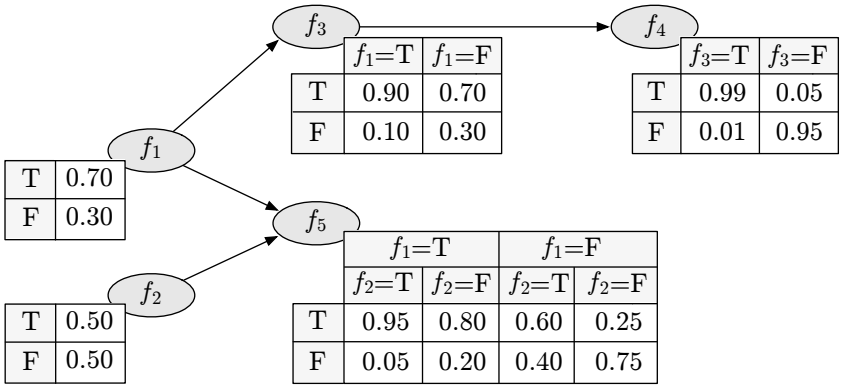


Figure 2.2: Abstract example of Bayesian network with boolean variables.

The *fundamental rule* for probability calculus is the following:

$$\mathsf{Pro}(A \mid B)\mathsf{Pro}(B) = \mathsf{Pro}(A, B),$$

where $\mathsf{Pro}(A, B)$ is the probability of the joint event $A \wedge B$. From this rule follows $\mathsf{Pro}(A \mid B)\mathsf{Pro}(B) = \mathsf{Pro}(B \mid A)\mathsf{Pro}(A)$ and this yields the *Bayes rule*:

$$\mathsf{Pro}(B \mid A) = \frac{\mathsf{Pro}(A \mid B)\mathsf{Pro}(B)}{\mathsf{Pro}(A)}$$

If A is a variable with values $\{v_1, \ldots v_n\}$ then $\mathsf{Pro}(A)$ is a probability distribution over these values:

$$\mathsf{Pro}(A) = (x_1, \ldots x_n) \qquad x_i \geq 0 \qquad \sum_{i=1}^{n} x_i = 1$$

where x_i is the probability of A assuming the value v_i. Notice that if A and B are variables, $\mathsf{Pro}(A, B)$ is the table of probabilities for the possible pairs of values of A and B. Finally, by marginalizing B out of $\mathsf{Pro}(A, B)$, we get $\mathsf{Pro}(A)$:

$$\mathsf{Pro}(A) = \sum_{B} \mathsf{Pro}(A, B)$$

Figure 2.3 provides an example of how the rules above are applied to calculate the probability distribution of a variable, namely f_3. In (a) we introduce the unconditional probability distribution of f_1 (the parent of f_3). In (b) we specify the $\mathsf{Pro}(f_3 \mid f_1)$, the conditional probability distribution of f_3. By applying the fundamental rule, we get (c) $\mathsf{Pro}(f_3, f_1)$, the joint probability for f_1 and f_3. Finally, (d) shows $\mathsf{Pro}(f_3)$ resulting from the marginalization of f_1 out of $\mathsf{Pro}(f_3, f_1)$.

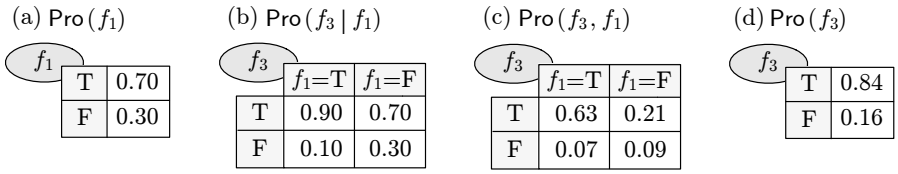


Figure 2.3: (a) Probability distribution for f_1, (b) conditional probabilities for f_3, (c) joint distribution for f_1 and f_3 and (d) probability distribution for f_3.

Inference with Bayesian networks is the task of calculating the posterior probability distribution for a set of variables, given some *evidences* indicating the value of other variables. Figure 2.4 provides an example, where *hard evidences*[4] on f_1 and f_2 are propagated throughout the network. The probability distributions of f_3, f_4 and f_5 have been computed by means of the rules previously introduced in this section.

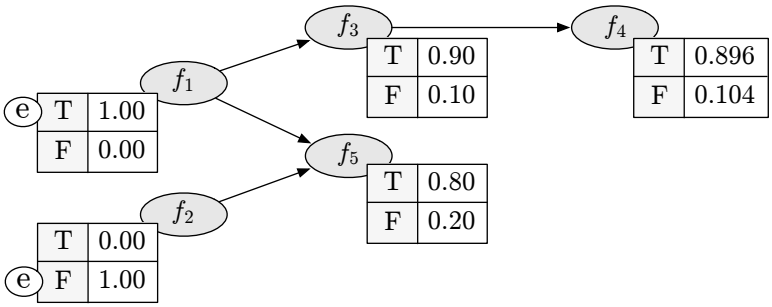


Figure 2.4: Bayesian network with two hard evidences.

The ways in which evidences may be propagated through a variable depend on its connections. Three connection types can be found in a Bayesian network:

(i) *Serial connections*: An evidence may be propagated through a serial connection unless the value of the variable in the middle is known. Thus, a hard evidence on the middle variable blocks the flow of information. For example, in Figure 2.2, consider the serially connected variables f_1, f_3 and f_4. If the value of f_3 is known, an evidence on f_1 does not affect the probabilities of f_4. Otherwise, if there is no evidence on f_3, the flow of information is free.

[4] An evidence that assigns zero probability to all except one value of a variable is called a *hard evidence* and it indicates that a particular value has been observed with certainty.

(ii) *Diverging connections*: An evidence may be propagated between all the children, unless the value of the parent node is known. In this case, a hard evidence on the parent node blocks the flow of information. For example, consider the diverging connection composed of f_1, f_3 and f_5 in Figure 2.2. If there is no evidence on f_1, then an evidence on f_5 may affect the probabilities of f_3.

(iii) *Converging connections*: An evidence may be propagated through a converging connection only if either the child node or one if its descendants has received evidence. This is the principle of *explaining away* which confirms and dismisses causes of events. For example, consider f_1, f_2 and f_5 in Figure 2.2. If there is an evidence on f_5, then an evidence on f_1 may affect the probabilities of f_2. Here, the variables f_1 and f_2 represent causes for f_5 and the knowledge of one cause may dismiss or confirm the other.

These three cases cover all the ways in which evidences may be propagated through a variable, and following these rules it is possible to determine for any pair of variables whether they are dependent given the observed evidences. Based on these rules, the *d-separation* of two variables is defined as follows.

Definition 3 (**d-separation**). *Two variables A and B are d-separated if for all paths between A and B there is an intermediate variable C, distinct from A and B, such that either the connection is serial or diverging and C has received evidence, or the connection is converging, and neither C nor any of its descendants have received evidence.*

If two variables A and B are d-separated with evidence e entered, then

$$\mathsf{Pro}(A \mid B, e) = \mathsf{Pro}(A \mid e),$$

This means that d-separation can be employed to determine conditional independencies, which allow us to compute prior and posterior probabilities more effectively.

There are several algorithms that automate the inference in Bayesian networks, such as the efficient message propagation inference algorithm for polytrees [91], the clique-tree propagation algorithm, also known as the *clustering* algorithm [72] and the variable elimination [124]. For a discussion about exact and approximate inference algorithms for Bayesian networks, see the survey by Guo and Hsu [58].

2.3 Normative multiagent systems

A *multiagent system*, as the name suggests, consists of multiple interacting *autonomous agents* within an environment [119]. This distributed paradigm has several advantages over centralized solutions. Multiagent systems can be utilized to distribute computational resources among interconnected agents, minimizing the problems with resource limitation and bottlenecks [96, 113], and to model problems in terms of autonomous interacting agents, which has demonstrated to be a more natural way of representing task allocation [106], team planning [77], simulations [118] and so on.

Nevertheless, these advantages do not come easily as autonomous agents might interfere with each other intentionally or due to a mere side-effect of their activities. Sometimes, these interferences have undesirable or disastrous consequences. In such situations, an agent might need to *coordinate* with their acquaintances [89, 43]. A basic approach to coordinating autonomous agents is to restrict their activities in a way which enables them to achieve their goals while not interfering with other agents [107]. In this sense, normative systems impose norms upon multiagent systems as an attempt to coordinate the agents' activities.

Boella et al. [14] define a *normative multiagent system* as follows:

> *A normative multiagent system is a multiagent system together with normative systems in which on the one hand agents can decide whether to comply with explicitly represented norms, and on the other the normative systems specify how and in which extent the agents can modify the norms.*

Agents are assumed to be *norm-autonomous*, that is, they can decide whether to comply with the norms. The way they take this decision is of crucial importance to this thesis, and Section 2.4 is dedicated to this subject. The study of agents that are capable of creating, changing and cancelling norms lie outside the scope of this thesis.

Explicit representations of the norms allow the agents to be informed about how they should behave (*regulations*) and commonly describe the sanctions against norm violations (*enforcement*) [13]. In order to provide explicit representations and computational interpretations to norms for use in the control of multiagent systems, several proposals of *normative languages* have been developed.

Vazquez-Salceda et al. [117] formalize a generic language for expressing conditions by means of a deontic logic with deadline operators. This logic includes obligations, permissions and prohibitions, possibly conditional, over actions or predicates (states). López [120] observes that the term norm has been used as a synonym for obligations [42], prohibitions [41], social laws [107], and other types of rules imposed by regulative mechanisms. Minding that these different definitions have common properties (social pressure, prescriptiveness and sociality), she represents them by using the same model encoded in Z language [110]. Kollingbaum and Norman [71] propose a language for the specification of normative concepts in norm-governed agents (this language is one of the components of the NoA agent architecture described in Section 2.4.5). Oren et al. [88] formalize an electronic contracting language, in which clauses within contracts are specified as permissions, obligations and prohibitions on contract parties. The norms in this language are associated with a status, which allows for the determination of whether contract violations takes place and who is the responsible for causing the violations. García-Camino et al. [53] propose a rule-based language to specify and manage permissions, prohibitions and obligations in electronic institutions [84, 44]. More concretely, this rule-based language is an extension of the general normative language proposed by Vazquez-Salceda et al. [117].

For a broader review of relevant research and open issues on normative multiagent systems, see the survey paper by Criado et al. [34].

2.4 Norm-aware agent models

This section makes a survey on normative agent models, where the autonomous agent research meets the ideas from deontic logic and traditional normative systems studied in philosophy and laws. Any normative reasoning approach outside of any particular agent model lies outside of the scope of this section.

Up to this moment, most research on normative reasoning by autonomous agents has been done from a practical reasoning perspective, through goal-oriented agent models. Such work proposes cognitive models for weighting competing alternatives (some of them non-compliant with the norms) on the basis of preference orders. Some approaches that apply normative reasoning with utility-based models have focused on the maximization of rewards, which is used as a quantitative measure of the profits and losses brought about by the adoption of a given set of norms.

2.4.1 Castelfranchi et al. (1999)

Deliberative normative agents are agents that have explicit knowledge about the enacted norms in a multiagent environment and can make a choice whether to obey the norms or not in specific cases. Castelfranchi et al. [28] propose a *generic* architecture for deliberative normative agents, which is able to know that a norm exists in a society, able to adopt that norm, able to deliberatively follow that norm, and able to deliberatively violate that norm in a intelligent way.

This architecture, designed as a refinement of the generic agent model proposed by Brazier et al. [22], includes components for reasoning in environments governed by norms. The main control has been refined into the following components: the *Norm Management* determines which norms the agent is adopting and in what ways the agent wants its behaviour to be influenced by norms; the component *Strategy Management*, therefore, uses norms to determine the strategies with which goals and plans are formed; on the basis of these strategies the component *Goal Management* determines which goals the agent wants to pursue, and the component *Plan Management* determines plans for the current goals of the agent.

To reflect semantic distinctions between different types of information within the agent, an object level and two meta-levels have been introduced. The information is processed according to its type through conditional statements (*if–then*) implemented in the components.

2.4.2 Boman (1999)

Boman [16] describes a method for enforcing norms onto *supersoft* agents programmed to represent and evaluate vague and imprecise information. These agents are assumed to behave in accordance with advices obtained from their individual *decision module*, with which they can communicate. Such a decision module contains algorithms for evaluating supersoft *decision data* concerning probability and utility.

Three ways of enforcing norm-compliant behaviour on the agents are proposed. The first one consists of manipulating the utilities in the lowest level to skew assessments of the *decision data* to have an overly positive or negative attitude towards some consequences. The second way consists of eliminating actions with disastrous consequences. The last way consists of disqualifying certain actions by referring to their negative impact on the global utility. This last option is a form of social norm adoption and it requires the knowledge regarding how to find the global utility values.

In fact, this work does not propose an agent model, but a method for enforcing norm-compliance onto a particular type of *utility-driven* agent. It assumes the existence of a decision module, whose advices are always followed by the agent. In other words, the agents are told what they must and must not do.

2.4.3 Dignum et al. (2000)

Dignum et al. [42] present an approach to social reasoning that integrates prior work in norms and obligations [114] with the BDI agent model developed by Rao and Georgeff [98]. Such an integration aims at introducing norms and obligations to support the socially motivated deliberation process of the agent. This type of agent has knowledge about the enacted norms and makes choices whether or not to obey norms, and how to weight up the impact of punishments for obligation violation in particular cases. The norms are not hard-wired into the agents: circumstances might change, making norms obsolete; and agents might interact with other agents that follow different norms, so explicit representation of norms and obligations can support a more flexible and appropriate reasoning.

Regarding the semantics of *norms* and *obligations*, it makes the following distinction between these concepts. Obligations are related to specific enforcement strategies which involve punishment of violators. They are explicit tools to influence the behaviour of the agent society. On the other hand, norms assist in standardizing the behaviour of the agents, making it easier to interact within that society. Norms have not explicit punishments and the consequences of failing to adhere to a norm can only be determined by considering a broader impact of indirect consequences.

The main control algorithm of the architecture is the same defined in [98], except that the selected events used in the *option-generator* are augmented with *potential deontic events* related to the applicability of norms and obligations. Constraints between the obligations are handled in the *option-generator*, which outputs a set of plans that can be executed simultaneously. In other words, each option contains a set of plans that are compatible among them. The *deliberation* process focuses on the selection of plans on the basis of *preferences*. The preference ordering of norms is based on a preference of social benefit of a situation, while the preference ordering of obligations are based on the punishments for violating them.

2.4.4 Broersen et al. (2002)

The BOID architecture [24] was proposed as a solution for BDI agents that act in a noisy environment, where the agent is overloaded with inputs. Its main problem is how an agent selects which obligation to comply with from a set of conflicting obligations, in addition to satisfying its own goals. BOID extends the BDI model by introducing obligations as a new component in addition to the main components of beliefs, desires and intentions, where goals are generated from the interaction between beliefs, obligations, intentions and desires, and biased by the type of agent: *realistic*, *stable*, *selfish* and *social*.

The agent's candidate goals are selected based on a static priority function on the rules that govern the agent's behaviour, which also determine which inference steps must be made. As a consequence, some rules may be overridden by others, enabling the resolution of conflict between mental attitudes by different agent types. If the agent's beliefs override its obligations, intentions or desires, then we say that this agent is *realistic*. If intentions override desires and obligations, then the agent is *stable*. If desires override obligations, then the agent is called *selfish*. Finally, if obligations override desires, then the agent is classified as *social*. Thus, BOID agents always consider norms in the same manner; that is, they cannot decide to follow or violate a given norm according to their circumstances.

The conflicts between mental attitudes can be classified into internal and external conflicts [23]. Internal conflicts are caused by information within the same component, such as the conflict between two beliefs or two obligations. External conflicts occur between information from two or more different components, such as a conflict between an intention and an obligation or between a desire and an obligation.

2.4.5 Kollingbaum and Norman (2003)

The Normative Agent Architecture [68], refereed to as NoA, aims to support the development of norm-motivated practical reasoning agents. NoA is based on classic BDI concepts with extensions that allow an agent to reason about norms. It is implemented as a reactive planning architecture composed of two main elements: the NoA *language* for the specification of plans and norms and the NoA *interpreter*, which is capable of interpreting and executing plan and norm specifications formulated in the NoA language.

The NoA language contains constructs for the specification of beliefs, goals, plans and norms. Norm specifications may regulate either the achievement of a particular state of affairs or the performance of explicit actions without consideration of the state that would produce. This particular feature is reflected in the plan specifications, which are characterised by explicit declarations of effects, where an effect may become the reason for a plan to be selected as an action. The interpreter, through the informed deliberation [71], allows the agent to remain *norm-autonomous* in that options for forbidden actions (or plans) which are not excluded, but are instead labelled as forbidden and remain options if an agent chooses to act in violation to resolve conflicts between norms. The consistency problem of norms in NoA architecture is detailed in [70, 69].

According to the authors, the NoA is influenced by the AgentSpeak(L) [97], but with three main distinctions. First, the NoA language allows *all* the effects of the plans to be declared. This provides greater flexibility in the specification of agent capabilities, and enables a NoA agent to reason about the side-effects of executing a plan. Second, the NoA agents are motivated by norms rather than desires and intentions. Third, it is based on the logic of responsibility for states and actions, and therefore captures the distinction between an agent taking responsibility for the achievement of a state of affairs and taking responsibility for the performance of an action.

2.4.6 Ågotnes et al. (2007)

Ågotnes et al. [1] develop a model of normative multiagent system in which the agents are assumed to have multiple goals of increasing priority. In this model, transitions are represented through Kripke structures and norms are implemented as constraints over these structures. A normative system is then simply a subset of the Kripke structure, which contains the arcs that are forbidden by the normative system.

The goals of the agents are specified as a hierarchy of formulae of Computational Tree Logic, which defines a model of ordinal utility, making possible the interpretation of the Kripke-based normative systems as games. Thus, the agents decide whether to defect or not from the normative system based on prioritized lists of goals and game theoretic solution concepts.

2.4.7 Andrighetto et al. (2007)

Andrighetto et al. [2] analyse inter agents and intra agent processes needed to deal with the emergence of norms. To do so, they put forward the EMIL-A, an agent architecture for recognising new norms and generating normative beliefs, deciding whether to adopt the norms, generating normative goals, determining whether to comply with them generating normative intentions, and finally, generating plans to achieve the normative intentions. In order to support the normative reasoning, there is also a normative long term memory containing a set of existing norms and normative information, and a repertoire of action plans consisting of norm-compliant actions.

EMIL-A agents are norm-autonomous: they can either comply with norms or violate them. Violations may, however, trigger defence mechanisms that are used to spread the norms to other agents. Apart from the architecture design, this work also focuses on the norm innovation process, showing how EMIL-A allows a new norm to be perceived and established as an instance of an existing norm, as part of the norm recognition component. Moreover, EMIL-A agents are also capable of determining the pertinence of norms and their degree of activation, that is, the norm salience. This norm salience is used as a criterion for accepting or rejecting norms, with the decision about norm compliance being determined by comparing the effects of violations with the costs of compliance.

2.4.8 López et al. (2007)

López et al. [123] present a formal normative framework for agent-based systems that includes a canonical model of norms, a model of normative multi-agent systems and a model of normative autonomous agents. In this paper, the authors put together the framework components, whose publication has been done in different forums.

According to the authors, norms are characterised by their *prescriptiveness* (it tells an agent how she should behave), *sociality* (situations where more than one agent is involved), and *social pressure* (socially acceptable mechanisms to force agents to comply with norms). Based on these properties, they define the components of a norm: *normative goals*, *addressee agents*, *beneficiaries*, *context* (circumstances in which the norm applies), *exceptions* (particular situations in which the addressees do not need to follow the norm), *rewards* and *punishments* (goals to be achieved by the agents entitled to do so). Some of these components can be used to classify the norms into four categories: *obligations*, *prohibitions*, *social commitments* and *social codes* [121].

To comply with a norm, the agent's deliberation process evaluates the set of goals that might be hindered by satisfying the normative goals, and set of goals that might benefit from the associated rewards. On the other hand, to reject a norm, an agent evaluates the damaging effects of punishments. The agents use the *importance* of their goals to make these decisions. There are different strategies to select the norms to be intended or rejected as explained in the paper [122] (*social*, *rebellious*, *pressured*, *opportunistic*). One example is the *pressured strategy* where an agent fulfils a norm only if one of its goals is threatened by punishments.

This normative framework provides the means to understand the normative behaviour of autonomous agents with explicit representation of their goals (BDI-like models). It assumes that these agents have a preference order over their goals, otherwise it would not be possible to take the decision regarding rejecting or complying with a given norm.

2.4.9 Meneguzzi and Luck (2009)

Assuming that agents are to operate in open environments, they need to adapt to changes in the norms that regulate such multiagent environments. In response, the paper [80] provides a technique to extend BDI languages by enabling them to enact behaviour modification at runtime in response to newly accepted norms.

To represent norms it proposes a schema, which contains the *norm* itself, its *activation condition* and *expiration condition*. The norm has two possible deontic modalities: *prohibition* and *obligation*. Both can refer to declarative world states or actions. The interpreter includes meta-level actions that allow an agent to scan its own plan library for plans that would violate a set of norms that an agent has previously accepted to comply. For prohibitions, violating plans are temporarily removed from the library while the prohibition is in effect. On the contrary, for obligations, new plans are created using a planning mechanism [79] so that an agent has plans that can accomplish such norms.

This work focuses on the problem of adapting the agent's behaviour to the *accepted* norms. The technique show in this work could be employed on the development of agents capable of violating norms, however it would demand the implementation of a process for deciding what norms are accepted or rejected.

2.4.10 Cardoso and Oliveira (2009)

Cardoso and Oliveira [26] implement norm-aware utilitarian agents characterized by different levels of *risk tolerance* and *social awareness*, which makes possible the generation of heterogeneous populations. The risk tolerance affects the decisions regarding opportunities to sign new contracts, and corresponds to the agent's willingness to contract in the presence of violation penalties. An agent decides to contract on the basis of the highest fine that is associated with the commitments for the assigned role. In order to contract, the following relation must be true:

$$highestFine(role) \leq b * RT/(1 - RT)$$

where b is a slope parameter related to the agent's budget and RT is the risk tolerance parameter. The social awareness, on the other hand, is related to decisions concerning ongoing contracts. This last parameter impels the agent to fulfil its obligations even when it does not have a strict advantage in doing so. An agent decides to fulfil an obligation o whenever the following relations is true:

$$violationOutcome(o) - fulfilmentOutcome(o) \leq b * SA/(1 - SA)$$

where b is a slope parameter related to the agent's budget and SA is the social awareness parameter.

2.4.11 Joseph et al. (2010)

Joseph et al. [64] formalize the principles of deductive coherence proposed by Thagard [112], and then, specifies a coherence-driven agent architecture which extends the BDI theory with the theory of coherence. Such an agent architecture takes decisions and actions, possibly non-normative, based on the coherence maximization. In other words, the agent obeys a norm only if it seems to be coherent.

Thagard's theory of coherence is the study of associations, that is, how a piece of information influences another and how best different pieces of information can fit together. Each piece of information imposes constraints on others, the constraints being positive (*coherence* degree) or negative (*incoherence* degree). Hence, a coherence problem is to put together those pieces of information that have a positive constraint between them, while separating those having a negative constraint. The pieces of information are represented as nodes in a graph with weighted links, or constraints,

between these nodes. Further, some of these constraints are positive and others negative, and associated with each constraint is a number that indicates the weight of the constraint. Therefore, maximising coherence is formulated as the problem of partitioning the set of nodes into accepted nodes and rejected nodes. The weight (strength) of a constraint is calculated by a deductive coherence function which captures the deductive relationship between propositions.

Based on the coherence framework, the authors propose a BDI architecture for coherence-driven agents. This agent architecture is an adaptation of the multi-context graded BDI model [27], on which the theories of the contexts will produce coherence graphs. For example, the belief theory τ_B gives rise to a coherence graph whose nodes are graded formulas of the belief language L_B used in the belief context. The coherence values between the belief nodes are determined not only by virtue of the deduction relationship, but also taking into account the grades (belief degree) as specified in the theory presentation τ_B. The same approach is used for the desire, intention and norm contexts.

In the norm context, a graded norm is interpreted in terms of its priority, which corresponds to a measure of its importance within a system of norms. To represent and reason with norms, the authors use the Probability-valued Deontic Logic [40], which do not represent of sanctions. In order to relate norm violations to sanctions, the coherence-driven agents use the implication operator: a proposition representing a violating state entails a proposition representing the effects of the respective sanction.

2.4.12 Oh et al. (2011)

Oh et al. [87] describe an agent for normative reasoning assistance that can proactively prevent human users from violating norms in a time-constrained environment. This assistant agent is composed of a *plan recognizer*, a *norm reasoner* and a *planner*. The plan recognizer, introduced in [86], identifies the user's needs in advance so the agent works in parallel with the user to ensure the assistance is ready by the time the user needs it. Based on the assumption that human users generally reason about consequences and take decisions to maximize their long-term rewards, the agent represents the planning problem as a Markov Decision Process (MDP) and uses an optimal policy to predict the future activities of the user.

Inspired by the normative structure proposed in [53], a norm is defined in terms of its *deontic modality*, its *context condition* specifying when a norm is relevant to a state, and its *normative condition* specifying the constraints imposed to the agent when the norm is relevant. If a given state is relevant to a norm, the normative condition is evaluated to determine the state's compliance, which depends on the deontic modality of the norm. Specifically, an obligation is violated if the normative condition is not satisfied in the given state, while a prohibition is violated if the normative condition is satisfied in the given state.

Given a predicted user plan in a plan-tree, the norm reasoner visits each node in the tree and evaluates the associated user state for any norm violations. Thus, the assistant can alert the user of active violations and proactively steer the user away from those violations that are likely to happen in the future. In order to accomplish this, for each state that violates a norm, the agent looks for the nearest state that is compliant with all norms. When a compliant state is found, this state becomes a new goal state for the agent, generating a planning problem to the planner module such that the agent needs to find a series of actions to move from initial state to this goal state.

2.5 Analyzing the norm-aware agent models

In order to compare the agent models described in Section 2.4, they are analyzed along five dimensions. Table 2.1 provides a summary of the outcome of this analysis, where the properties accounted for in the respective agent architectures are checked.

- *Norm-autonomy*: This dimension determines whether an agent is capable of violating norms, that is, norm-autonomous. On the other hand, an agent with no autonomy with respect to norms does not exercise control over its own actions that are regulated by the norms.

- *Explicitness of norms*: Concerns to the property of having norms represented as some explicit normative structure, which allows mirroring changes in the norms regulating the environment.

- *Explicitness of sanctions*: This property can be found in an agent model if it explicitly represents sanctions and takes them into account for deciding whether to comply with the norms;

- *Quantitative decision model*: This dimension refers to architectures that make decisions *exclusively* on the basis of quantitative information, namely expected utilities, obtained from certain action and/or states of the world. On the other hand, goal-oriented agent models, usually use some preference orderings related to the agents' mental states. Some of these models associate utilities with goals, however, it does not necessarily mean that the goals are selected exclusively on the basis of their utilities. Usually, these goal-oriented agents run a deliberation process that considers sources of qualitative information to decide which goals to pursue.

- *Probabilistic planning*: Concerns techniques that an autonomous agent can use to choose actions in the face of uncertainty about its environment and the results of its actions.

The generic architecture proposed by Castelfranchi et al. [28] introduces the idea of useful norm violations, highlighting the importance of not hard-wiring norms into the agents' code, and discusses principles for agents that are able to behave deliberatively on the basis of explicitly represented norms. According to the authors, the precise knowledge by which goals are generated depends on the application addressed; on this basis, this generic normative agent model does not commit to a specific decision model or planning algorithm.

In Boman [16], norms are used as constraints to filter the agent's alternatives to disable any deviant behaviour. That is, the agent has no autonomy to violate norms. Furthermore, norms and sanctions are not explicitly represented. To decide between norm-compliant options, this agent model uses supersoft decision data concerning probability, utility, credibility and reliability, and to construct plans, it formulates exclusive and exhaustive action strategies.

Dignum et al. [42] focus on the development of socially responsible norm-autonomous BDI agents which adopt norms in an attempt to support collaborative behaviour. Norms are explicitly represented using a preference-based dyadic deontic logic (PDL) [114], and the choices between conflicting norms are made with predefined preference orderings. Sanctions, on the other hand, are not explicitly represented; instead, they are encoded in the function that computes the social worth of the situations. In order to achieve normative and personal goals, this agent architecture selects plans from a library based on meta-plans or hard-wired strategies.

Table 2.1: Related work properties

Work reference	**Norm-autonomy**	**Explicitness of norms**	**Explicitness of sanctions**	**Quantitative decision model**	**Probabilistic planning**
Castelfranchi et al. [28]	✓	✓			
Boman [16]				✓	✓
Dignum et al. [42]	✓	✓			✓
Broersen et al. [24]	✓	✓			
Kollingbaum and Norman [68]	✓	✓			
Ågotnes et al. [1]	✓	✓		✓	
Andrighetto et al. [2]	✓	✓		✓	
López et al. [123]	✓	✓	✓		
Meneguzzi and Luck [80]	✓	✓			
Cardoso and Oliveira [26]	✓	✓	✓	✓	
Joseph et al [64]	✓	✓	✓		
Oh et al. [87]		✓		✓	✓

The compliance with norms in the BOID architecture [24] depends on the establishment of commitments to the respective normative goals, which are generated from explicit representations of the norms. The agent's candidate goals are selected based on a static priority function on the rules that govern the agent's behaviour. The authors briefly discuss the impact of obligations in the planning and scheduling, however the specification of these tasks are beyond the scope of this work.

NoA [68] is one of the first practical architectures to support the implementation of norm-governed practical reasoning agents. This architecture is based on classic BDI models, with extensions that allow an agent to reason about norms. It takes influences from classical planners with respect to the declaration of plans – the behaviour of the agent is determined by pre-specific plans composed of deterministic actions. In this model, punishments for norm violations are not taken into account – the agents aim at maximizing the consistency level of the adopted norms, and norm violations take place only in circumstances where compliance is not possible.

Ågotnes et al. [1] account for strategic behaviour by agents when deciding whether to comply with the normative system or not. To do so, the authors use an ordinal utility which allows the interpretation of their Kripke-based normative system as games, in which agents determine whether to follow the norms. In this work, norms are limited to prohibitions, which are explicitly represented as forbidden transitions in the Kripke structure, and sanctions are not taken into account. A scenario with sequential decision making is not considered as well.

Through EMIL-A [2], Andrighetto et al. explain the phases that norms undergo so as to evolve from the environment into the internal state of norm-autonomous agents. The decision about norm compliance is determined by the expected utility that agents should obtain if they fulfil or violate the norm, taking into account the penalties and incentives associated to the normative frame. However, this agent model, as shown in [2], neither includes an explicit representation of sanctions nor details the normative action planner.

Similarly to the BOID architecture, the autonomous normative agent architecture proposed by López et al. [122] adopts the notion of stereotypes (i.e. opportunistic, social, rebellious), which are mapped to goal selection strategies ruled by priority functions. This work proposes three normative reasoning processes (none of them related to planning): one for agents to decide whether to adopt a norm, another to decide whether to comply with a norm, and the other to update the goals.

The technique proposed by Meneguzzi and Luck [80] focuses on the problem of adapting the agent's behaviour to the accepted norms, letting outside the scope the module for choosing which norms are accepted or rejected. The authors demonstrate the viability of their approach through an implementation in the AgentSpeak(L) [97], a language to develop BDI agents, which uses first-order logic to represent beliefs, with which is not possible to represent probabilistic information.

Cardoso and Oliveira [26] study adaptive mechanisms that enable a normative framework to change its deterrence sanctions depending on the population. The agents developed in that work do not construct plans or policies, but are utility maximizers defined by two parameters: social awareness and risk tolerance. Based on these parameters, they decide whether to violate a norm or not.

Joseph et al. [64] adapt the multi-context BDI model [27], which accounts for representing non-deterministic information through grades. Obligations are represented through propositions in a graded normative language, while sanctions are represented via the implication operator: a violation entails a proposition representing the effects of the sanction (in the context of this discussion, both have been considered explicit representations). The norm adoption process is made by maximizing the coherence between mental attitudes in the place of maximizing expected utilities.

Oh et al. [87] describe an agent for normative reasoning assistance that is capable of proactively preventing human users from violating norms. The norms are explicitly represented through obligations and prohibitions. As the norms are addressed to the human users, the assistant agent cannot be considered norm-autonomous. The agent is deployed in a time-constrained collaborative environment with no sanctions.

2.6 Discussion

This chapter has provided some relevant literature and background for this thesis. Starting with the MDP theory, we have covered computational methods to construct optimal policies, approaches for achieving computational leverage, and generalizations of the MDP framework to address several problems. Among these proposals to solve coordination problems and for achieving computational leverage, none have exploited *norms* for these purposes.

In analyzing the norm-aware agent models, we have studied several proposals for enabling agents to reason about norms. The analysis has shown that no agent model has covered all the analyzed dimensions, and only three models are capable of planning with probabilistic information.

Clearly, there is *gap* between the advances in *normative multiagent systems*, and the generalizations and computational methods for Markov Decision Processes. By bridging this gap, each research area could benefit from the native strengths of the other. Norms could help with coming up with a way to shape the agents' behavior, enabling coordination and improving the tractability of MDPs by restricting the state space to be explored. On the other side, MDPs can help with their ability to perform quantitative evaluations, which in this case, could be used to evaluate the impact of norms in the agents' utilities.

Part II

NMDP framework and agent models

Chapter 3

Normative MDP framework

Simplicity is prerequisite for reliability.

Edsger Dijkstra

As we have seen in Chapter 2, there have been several proposals for enabling agents to reason about norms, acknowledging the advantages of norms as a means to condition agent behavior. However, the emphasis in these proposals has been on the feasibility of such normative reasoning rather than on the ability to quantitatively reason about agent performance, especially in the presence of uncertainty. In Chapter 2, we also have seen that there have been some proposals to constrain the search space of policies of MDPs in order to tackle coordination problems or get computational leverage, but none of these proposals have employed solutions based on normative systems.

The proposal for modeling norm-aware rational agents put forward in this chapter combines the native strengths of the MDPs and norms in order to cover their individual limitations. It combines, on one hand, the analytic advantages of MDPs to model the domain of interaction and agent decision-making under uncertainty, and, on the other, it uses normative structures to support coordination and computational leverage techniques.

This chapter introduces the *Normative Markov Decision Process* (NMDP) framework, a proposal for modelling norm-aware rational agents acting in non-deterministic environments. Section 3.1 formalizes the NMDP framework utilizing set theoretical notions to define norms and detection models of norm violations. Section 3.2 deals with the identification of properties of norms, such as conflicts between norms and conflicts between sanctions. Section 3.3 shows how factored state spaces can be exploited in order to compactly encode norms and detection models. Finally, Section 3.4 closes this chapter by promoting a discussion about the NMDP framework.

3.1 Normative Markov Decision Processes

The NMDP framework extends the well-known MDPs, broadly utilized for modeling sequential decision making of single agents in stochastic domains, in order to explicitly represent norms, sanctions and detection probabilities of norm violations. The MDP framework has been extended by two additional components: $\mathcal{N}$ denotes a set of norms ruling system, and $\mathcal{D}$ denotes a detection probability function of norm violations.

Definition 4 (**Normative Markov Decision Process**). *A* NMDP *is a tuple* $\langle \mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{T}, \mathcal{R}, \mathcal{N}, \mathcal{D} \rangle$ *where:* $\mathcal{N}$ *is a set of norms designed to regulate a group of agents in the system;* $\mathcal{D} : \mathcal{N} \times \mathcal{S} \rightarrow \mathbb{R}$ *is a detection function, also referred to as detection model* ($\mathcal{D}(q, s)$ *indicates the detection probability of the violation of the norm* $q \in \mathcal{N}$ *in the state* $s \in \mathcal{S}$)*; the remaining components are inherited from the* MDP *framework.*

Several normative languages have been proposed in previous research on normative multiagent systems, being the majority of these proposals established on deontic notions [117, 123, 53]. In the present work, norms are described using set theoretical notions. A norm is described as a set of states that are *prohibited* or *obliged* for a group of agents. So, depending on its deontic modality, a norm is classified into *obligation* or *prohibition*. A norm has a *context* (set of states) where it applies. Any state outside the context is irrelevant to the norm. The obliged or prohibited states are determined by the *content* of the norm, which corresponds to a subset of states of the context. A *sanction* consists of a penalty and an enforced state-transition aiming at updating the current state of the addressee agents. The underlying intention of these sanctions is to punish the transgressors by decreasing their utility, and moving them from violating states to states where the norms are obeyed and/or their capabilities are constrained.

Definition 5 (Norm). *A norm is a tuple* $\langle \alpha, \delta, \mathcal{G}, \mathcal{X}, \mathcal{E}, \sigma \rangle$ *where the following constraints hold:*

- $\alpha \in \mathbb{N}$ *is an unique identifier denoting the priority of the norm, such that* $1 \leq \alpha \leq |\mathcal{N}|$ *. Here,* $\alpha{=}1$ *is the highest priority and* $\alpha{=}|\mathcal{N}|$ *is the lowest.*
- $\delta \in \{$*obligation, prohibition*$\}$ *is the deontic modality.*
- $\mathcal{G}$ *is the group of agents to which the norm applies.*
- $\mathcal{X} \subseteq \mathcal{S}$ *is the set of states composing the normative context where the norm applies.*
- $\mathcal{E} \subseteq \mathcal{X}$ *is the set of states contained in the normative context which are obliged or prohibited depending on the deontic modality.*
- σ *is a sanction represented by a tuple* $\langle \rho, \phi \rangle$ *where:*
  - $\rho : \mathcal{S} \rightarrow \mathbb{R}$ *is a function that provides the penalty for violating this norm in a given state (* $\rho(s)$ *gives the penalty to be paid in* s*).*
  - $\phi : \mathcal{S} \rightarrow \mathcal{S}$ *is a function that calculates the outcome state resulting from an enforced state-transition in response to the violation of this norm (*$\phi(s)$ *gives the outcome of an enforced transition in* s*).*

A norm is read as follows: if the norm is a *prohibition*, in the set of states $\mathcal{X}$ where the norm applies, any agent in the group $\mathcal{G}$ is prohibited to be in any state in $\mathcal{E}$; if the norm is an *obligation*, in the set of states $\mathcal{X}$ where the norm applies, any agent in the group $\mathcal{G}$ is obliged to be in some state in $\mathcal{E}$. As we can see, depending on the deontic modality, a norm designates prohibited or obliged states. If the agent is sanctioned in response to the violation of the norm, this agent is penalized with $\rho(s)$ and moved to the state $\phi(s)$, where s is the state where the norm violation took place. Example 1 provides an example of norm in the traffic domain.

Example 1. *Suppose a norm that obligates a travel direction in a given street. If an agent violates this norm by traveling in the opposite direction of the one prescribed by the norm, then this agent receives a fine, apart from being forced to turn around and go to the norm-compliant direction, or in some situations, even losing its license. This obligation is addressed to a group of agents* $\mathcal{G}$ *that are drivers. Clearly, the context* $\mathcal{X}$ *of this norm consists of all states representing the given street, and the content* $\mathcal{E}$ *of this*

norm consists of all states where the agent is traveling in the prescribed direction. The sanctions are implemented with functions from violating states to penalty values, and from violating states to outcome states in the case of the enforced state-transitions.

Definition 6 (**Prohibited state**). *Let $s \in \mathcal{S}$ be a state, $q \in \mathcal{N}$ be a prohibition denoted as $\langle \alpha, \textit{prohibition}, \mathcal{G}, \mathcal{X}, \mathcal{E}, \sigma \rangle$. The state s is prohibited by q iff $s \in \mathcal{E}$.*

Definition 7 (**Obliged state**). *Let $s \in \mathcal{S}$ be a state, $q \in \mathcal{N}$ be an obligation denoted as $\langle \alpha, \textit{obligation}, \mathcal{G}, \mathcal{X}, \mathcal{E}, \sigma \rangle$. The state s is obliged by q iff $s \in \mathcal{E}$.*

A set of states is relevant to a norm $\langle \alpha, \delta, \mathcal{G}, \mathcal{X}, \mathcal{E}, \sigma \rangle$ if this set of states is a subset of $\mathcal{X}$, which indicates the context where the norm applies. Given a set of states that are relevant to a norm, we can determine which of them violate or comply with it.

Definition 8 (**Set of states that violate a norm**). *Given a norm $q \in \mathcal{N}$ denoted as $\langle \alpha, \delta, \mathcal{G}, \mathcal{X}, \mathcal{E}, \sigma \rangle$, the set of states of an NMDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{T}, \mathcal{R}, \mathcal{N}, \mathcal{D} \rangle$ that violate q, denoted as $\mathcal{S}_q^{\triangledown}$, is defined as:*

$$\mathcal{S}_q^{\triangledown} = \begin{cases} \mathcal{E} & if \quad (\delta = \textit{prohibition}) \\ \mathcal{X} \setminus \mathcal{E} & if \quad (\delta = \textit{obligation}). \end{cases}$$

That is, for a prohibition, every prohibited state is norm violating. On the other hand, for an obligation, a state is violating if it is in the normative context but it is not obliged. The set of states that comply with the norm $q = \langle \alpha, \delta, \mathcal{G}, \mathcal{X}, \mathcal{E}, \sigma \rangle \in \mathcal{N}$, denoted as $\mathcal{S}_q^{\diamondsuit}$, consists of those states in the normative context $\mathcal{X}$ which do not violate the norm q, that is:

$$\mathcal{S}_q^{\diamondsuit} = \mathcal{X} \setminus \mathcal{S}_q^{\triangledown}.$$

It is possible to define the set of norm violating states, which contains all states that violate some norm. This definition is useful to define norm-compliant behaviours, which avoid violating states regardless of the norms that they violate.

Definition 9 (**Norm violating states**). *The set of violating states of an NMDP is composed of all states in which at least one norm in $\mathcal{N}$ is violated:*

$$\mathcal{S}^{\triangledown} = \bigcup_{q \in \mathcal{N}} \mathcal{S}_q^{\triangledown}.$$

From the relative complement of the set of norm violating states in the entire state space, we obtain the set of norm-compliant states, denoted as $\mathcal{S}^{\Diamond}$, which contains the states in $\mathcal{S}$ that comply with all norms in $\mathcal{N}$:

$$\mathcal{S}^{\Diamond} = \mathcal{S} \setminus \mathcal{S}^{\nabla}.$$

Regarding the detection model, it is assumed that detected violations are sanctioned, so, the function $\mathcal{D}(q, s)$ provides the measure of the rate at which the sanction of q is executed if q is violated in the state s. For example, if the violation of the norm in Example 1 is detected with probability 0.1 in all states, then the agent goes unpunished with probability 0.9.

3.2 Properties of norms

Norms provide a useful abstraction with which restrictions can be addressed to the agents. An important aspect to be taken into account during the design of the norms is that they may conflict with one another. In such situations, an agent has no option of complying with the norms given that whatever they do causes a norm violation.

Definition 10 (**Conflicting norms**). *Two norms, denoted as q_i and q_j, are in conflict in a given state $s \in \mathcal{S}$ iff this state simultaneously complies with one norm and violates the other, formally:*

$$\left(s \in \mathcal{S}^{\nabla}_{q_i} \wedge s \in \mathcal{S}^{\Diamond}_{q_j}\right) \vee \left(s \in \mathcal{S}^{\nabla}_{q_i} \wedge s \in \mathcal{S}^{\Diamond}_{q_j}\right).$$

In this case, the given state violates one norm and complies with the other, which makes impossible to follow both norms in this state. Example 2 provides examples of conflicting norms, showing that conflicts can happen between norms with the same deontic modality or between norms with different deontic modalities.

Example 2. *Suppose two norms that regulate the travel direction in the same street. The following items describe the possible conflicts between these norms, which let the agents with no option of complete norm-compliance given that no state satisfy both norms simultaneously:*

(*i*) *These norms obligate opposite travel directions.*

(*ii*) *Together, these norms prohibit all travel directions.*

(*iii*) *One norm obligates a particular travel direction, and the other prohibits it.*

If the entire state space if free of normative conflicts, then we say that the set of norms is *normative consistent*.

Definition 11 (**Normative consistency**). *A set of norms $\mathcal{N}$ is consistent if there is no state in $\mathcal{S}$ that presents a conflict between norms.*

Another aspect to be considered in the design of norms is the existence of conflicts between sanctions, given that in such circumstances, the imposition of conflicting sanctions affects the outcome of each other. Therefore, the outcome of the enforced-state transitions depends on the order of execution of the conflicting sanctions.

Definition 12 (**Conflicting sanctions**). *Two sanctions, denoted as $\langle \rho_i, \phi_i \rangle$ and $\langle \rho_j, \phi_j \rangle$, are in conflict in a given state $s \in \mathcal{S}$ iff the functions ϕ_i and ϕ_j are not permutable*[1]*:*

$$\phi_i(\phi_j(s)) \neq \phi_j(\phi_i(s)).$$

Example 3 provides an example of sanctions which intend to bring about mutually exclusive outcomes. In this example, it is assumed that the last sanction to be executed cancels the outcome of the other.

Example 3. *Suppose that two traffic norms are violated in a given state, where the sanction of the first norm determines that the permanent license of the agent must be suspended, and the sanction of the second norm determines that the license must be downgraded to probationary. There is a conflict of sanctions here, given that there is no state where the license is simultaneously suspended and probationary.*

Notice that this section aims exclusively at the identification of conflicts between sanctions. Their resolution is discussed in Chapter 4, where the normative reasoning problem is approached with the NMDP framework.

[1] According to Ritt [100], two rational functions f and g are *permutable* if $f(g(x)) = g(f(x))$. This thesis uses the same definition, however, with functions that calculate the outcome resulting from enforced state-transitions in response to norm violations.

3.3 Compact representation of NMDPs

Previous research, as shown in the overview by Boutilier et al. [19], has demonstrated that *factored* representations for state spaces can be used to design efficient MDP solution techniques that exploit a certain *structure* in the state space. The *factored* approach[2], differently from an *extensional* representation, structures the state space explicitly through *factors* or *features*. This section shows how factored state spaces can be exploited to construct compact representations of norms and detection models.

3.3.1 Factored state space

Let $\mathcal{F}$ be the set of all features, a feature $f_i \in \mathcal{F}$ takes on a finite number of values, and $\mathcal{V}_{f_i}$ stand for the finite set of possible values of f_i. A state is any possible assignment of values to these features, and the state space $\mathcal{S}$ is the cross product of the value spaces for the features as follows:

$$\mathcal{S} = \times_{i=1}^{F} \mathcal{V}_{f_i}$$

Table 3.1 provides an example of factored state space of an agent in an urban road network made up of streets composed of discrete cells. In this network, the position of agent is defined by Street and Cell, and its orientation is defined by Direction. The Status of the agent indicates its present situation, which can be holding a position, moving through the network or ended. The License indicates the type of driving license that the agent possesses, and finally, Seatbelt indicates if the agent's seatbelt is fastened or not.

Table 3.1: Example of factored state space.

i	f_i	$\mathcal{V}_{f_i}$
1	Street	{00, 01, ... 07}
2	Cell	{00, 01, ... 23}
3	Direction	{Up, Right, Down, Left}
4	Status	{Still, Moving, Ended}
5	License	{Permanent, Probation, Suspended}
6	Seatbelt	{Fastened, Unfastened}

[2]Factored MDPs, as defined in [19], use compact representations for the transition model. However, the work developed in this thesis uses an extensional representation of transitions, focusing this way on the factored representation of the new components $\mathcal{N}$ and $\mathcal{D}$.

3.3.2 Norms

In factored state spaces, the states in $\mathcal{X}$ and $\mathcal{E}$ can be described by a logical formula ξ, over the features. The syntax of this formula can be defined according to the following grammar:

$$\mathsf{NC} ::= \neg\mathsf{NC} \mid (\mathsf{NC} \vee \mathsf{NC}) \mid (\mathsf{NC} \wedge \mathsf{NC}) \mid \mathsf{FV}$$
$$\mathsf{FV} ::= (\ feature = value\ )$$

where $value \in \mathcal{V}_{feature}$ is the value assigned to $feature$; $\neg$ is the negation operator; and $\vee$ and $\wedge$ are the connectives disjunction and conjunction. In the definition of $\mathcal{X}$, this formula is called *normative context*, and in the definition of $\mathcal{E}$, it is called *normative content*.

The procedure of verifying if a state is relevant to a norm constitutes a boolean satisfiability problem on which we aim at determining if the features defining the state satisfy the normative context. Linking this feature-based approach with the set theoretical Definition 4, $s \in \mathcal{X}$ if s satisfies the normative context formula, and $s \in \mathcal{E}$ if s satisfies the normative content formula.

Definition 13 (**Satisfiability**). *A state $s \in \mathcal{S}$ satisfies a formula ξ (denoted as $s \vDash \xi$) if the value of the features defining the state s can be assigned to ξ, in such a way as to make this formula evaluate to true.*

To find out if a state $s \in \mathcal{S}$ complies or violates a norm $q \in \mathcal{N}$, we first have to determine if q applies in s. Let $\xi_{\mathcal{X}}$ be the formula specifying the normative context of q; if $s \vDash \xi_{\mathcal{X}}$ then q applies in s. In this case, we determine if s satisfies $\xi_{\mathcal{E}}$, the formula specifying the content of q. If $s \vDash \xi_{\mathcal{E}}$ then s is obliged or prohibited, depending on the deontic modality of q. Then, the compliance of q in s is determined as follows:

- if ($\delta = \textit{obligation}$), ($s \vDash \xi_{\mathcal{X}}$) and ($s \vDash \xi_{\mathcal{E}}$) then s complies with q.
- if ($\delta = \textit{obligation}$), ($s \vDash \xi_{\mathcal{X}}$) and ($s \nvDash \xi_{\mathcal{E}}$) then s violates q.
- if ($\delta = \textit{prohibition}$), ($s \vDash \xi_{\mathcal{X}}$) and ($s \vDash \xi_{\mathcal{E}}$) then s violates q.
- if ($\delta = \textit{prohibition}$), ($s \vDash \xi_{\mathcal{X}}$) and ($s \nvDash \xi_{\mathcal{E}}$) then s complies with q.
- if ($s \nvDash \xi_{\mathcal{X}}$) then q is not relevant in s.

In order to specify the penalty function ρ, an ordered set of rules was used, which is defined by the following grammar:

$$\begin{aligned}
\mathsf{RULESET} &::= \{\, \mathsf{RULE}, \mathsf{DEFRULE} \,\} \mid \{\, \mathsf{DEFRULE} \,\} \\
\mathsf{RULE} &::= \mathsf{RULE}, \mathsf{RULE} \mid \mathsf{RC} \rightarrow \mathit{penalty} \\
\mathsf{DEFRULE} &::= \top \rightarrow \mathit{penalty} \\
\mathsf{RC} &::= \neg \mathsf{RC} \mid (\mathsf{RC} \vee \mathsf{RC}) \mid (\mathsf{RC} \wedge \mathsf{RC}) \mid \mathsf{FV} \\
\mathsf{FV} &::= (\ \mathit{feature} = \mathit{value}\ )
\end{aligned}$$

where $\mathit{value} \in \mathcal{V}_{\mathit{feature}}$ is the value assigned to $\mathit{feature}$, and $\mathit{penalty} \in \mathbb{R}$. In this grammar, a rule has the form $\omega \rightarrow \lambda$, where ω is a condition represented by a logical formula and λ is a penalty. To obtain the penalty for an input state $s \in \mathcal{S}$, the applicability of the rules is checked sequentially until this state satisfies the condition of some rule. If $s \vDash \omega$ then λ is the penalty to be paid in s. The default penalty is provided by the last rule $\top \rightarrow \lambda$ whose condition is always interpreted as true.

The same rule-based approach is used in the specification of the function ϕ, whose rules are defined as follows:

$$\begin{aligned}
\mathsf{RULESET} &::= \{\, \mathsf{RULE}, \mathsf{DEFRULE} \,\} \mid \{\, \mathsf{DEFRULE} \,\} \\
\mathsf{RULE} &::= \mathsf{RULE}, \mathsf{RULE} \mid \mathsf{RC} \rightarrow \{\, \mathsf{FEATLIST} \,\} \\
\mathsf{DEFRULE} &::= \top \rightarrow \{\, \mathsf{FEATLIST} \,\} \\
\mathsf{RC} &::= \neg \mathsf{RC} \mid (\mathsf{RC} \vee \mathsf{RC}) \mid (\mathsf{RC} \wedge \mathsf{RC}) \mid \mathsf{FV} \\
\mathsf{FEATLIST} &::= \mathsf{FEATLIST}, \mathsf{FEATLIST} \mid \mathsf{FV} \\
\mathsf{FV} &::= (\ \mathit{feature} = \mathit{value}\ )
\end{aligned}$$

where $\mathit{value} \in \mathcal{V}_{\mathit{feature}}$ is the value assigned to $\mathit{feature}$. In this grammar, a rule has the form $\omega \rightarrow \{\theta_1, \ldots, \theta_n\}$, where ω is a condition represented by a logical formula and θ_i, such that $1 \leq i \leq n$, is a feature with an assigned value. In order to obtain the outcome state resulting from an enforced transition in an input state $s \in \mathcal{S}$ (to determine $\phi(s)$), the applicability of the rules is checked sequentially. When $s \vDash \omega$, the list of features $\{\theta_1, \ldots, \theta_n\}$ is used to update s by overwriting the value of some of its features, obtaining in this way the outcome state of an enforced state-transition in s. Again, the default outcome state is given by the last rule $\top \rightarrow \{\theta_1, \ldots, \theta_n\}$.

Example 4 shows how the factored state space specified in Table 3.1 can be used in the specification of a norm regulating the traffic direction.

Example 4 (revisiting Example 1)**.** *As an example of normative structure instance, assume an obligation indicating that agent* 01 *on street* 00 *must be heading right. The sanction specifies that if the agent's license is suspended, it pays* -5.0 *and is ended. If the agent is a probationary driver, it pays* -2.5 *and its license is suspended. Finally, in the remaining cases, the agent pays* -1.0 *and turned to the right direction.*

$$\begin{aligned}
&\langle 1, \textsc{Obligation}, \{\textsc{Agent}\ 01\},\\
&\quad (\textsc{Street}{=}00), (\textsc{Direction}{=}\textsc{Right}),\\
&\quad\quad \langle\ \{\ (\textsc{License}{=}\textsc{Suspended}) \rightarrow -5.0,\\
&\quad\quad\quad (\textsc{License}{=}\textsc{Probation}) \rightarrow -2.5,\\
&\quad\quad\quad \top \rightarrow -1.0\ \},\\
&\quad\quad\ \{\ (\textsc{License}{=}\textsc{Suspended}) \rightarrow \{(\textsc{Status}{=}\textsc{Ended})\},\\
&\quad\quad\quad (\textsc{License}{=}\textsc{Probation}) \rightarrow \{(\textsc{License}{=}\textsc{Suspended})\},\\
&\quad\quad\quad \top \rightarrow \{(\textsc{Direction}{=}\textsc{Right})\}\ \}\ \rangle\rangle
\end{aligned}$$

Example 5 provides an example of normative conflicts in a given state using a feature-based representation.

Example 5 (revisiting Example 2)**.** *Suppose two norms* (q_i *and* q_j) *that intend to regulate the traffic in the context of* (Street=00)*, which admits two travel directions: from left to right or from right to left. Let* $s \in \mathcal{S}$ *be a state that* $s \vDash$ (Street=00)*, so* s *is part of the context of* q_i *and* q_j*. These norms are in conflict in* s *if:*

(*i*) q_i *obligates* (Direction=Right) *and* q_j *obligates*(Direction=Left),
q_i *obligates* (Direction=Left) *and* q_j *obligates*(Direction=Right),
q_j *obligates* (Direction=Right) *and* q_i *obligates*(Direction=Left), *or*
q_j *obligates* (Direction=Left) *and* q_i *obligates*(Direction=Right).

(*ii*) q_i *prohibits* (Direction≠Right) *and* q_j *prohibits* (Direction≠Left).

(*iii*) q_i *prohibits* (Direction=Right) *and* q_j *obligates* (Direction=Right),
q_i *prohibits* (Direction=Left) *and* q_j *obligates* (Direction=Left),
q_j *prohibits* (Direction=Right) *and* q_i *obligates* (Direction=Right), *or*
q_j *prohibits* (Direction=Left) *and* q_i *obligates* (Direction=Left).

In this feature-based approach, a conflict of sanctions occurs in a given state if there are sanctions updating the same feature with different values. In this case, the final value of this feature is determined by the order of application of the functions ϕ: the last function invoked overrides the value assigned by the previous. If two sanctions $\langle \rho_i, \phi_i \rangle$ and $\langle \rho_j, \phi_j \rangle$ are in conflict in a given state $s \in \mathcal{S}$ and the first has priority over the second, then the correct order of application is $\phi_i(\phi_j(s))$. Example 6 provides an example of sanctions in conflict.

Example 6 (revisiting Example 3)**.** *Suppose that an agent, currently possessing a permanent license* (LICENSE=PERMANENT)*, simultaneously violates two norms whose sanctions are in conflict. The sanction of the first norm determines the suspension of the agent's license* (LICENSE=SUSPENDED)*, while the sanction of the second norm determines that the agent must be placed on probation* (LICENSE=PROBATION)*.*

3.3.3 Detection model

One way of representing the detection probabilities consists of creating a lookup table with three columns, where the first column specifies the norm q, the second column specifies the state s and the third column provides the probability of detecting the violation of the norm q in the state s. In this *extensive* approach, $\mathcal{D}(q, s)$ returns the corresponding probability after finding it in the table.

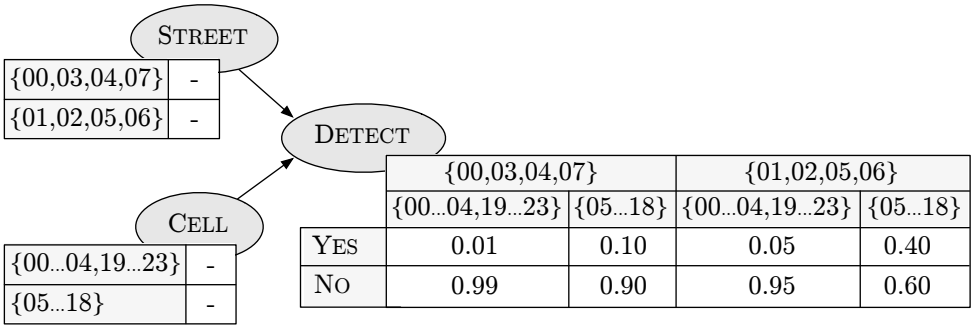
Fortunately, many large state spaces have significant internal structure, and often, the probability of detecting the violation of a norm depends only on a small number of features of the state space. So, as an alternative to the extensive approach, it is possible to utilize Bayesian networks [92, 63], which exploit the structure (factorization) of the state space to construct a compact representation of the detection model. For instance, the probability of some policeman to detect the violation of the norm shown in Example 4 is likely to depend on the city region (the concentration of police forces in the city center is likely to be higher than in a quiet neighbourhood, so wrong-way driving in these regions may have different detection probabilities), but it is very unlikely to depend on the driver's license type.

A Bayesian network expressing the probabilities of detecting the violation of a norm (or set of norms) is composed of n parent nodes, each of them representing a feature of the state space that is relevant to the prediction of the probabilities (features not represented are assumed to be irrelevant), and one child node that provides the detection probabilities. The Conditional Probability Table (CPT) of the child node

contains the probabilities of detecting the norm violation given the conditions imposed by the parent nodes. These conditions are set by hard evidences (certain observations of feature values) on the parent nodes, indicating the state for which we intend to calculate the detection probability. Based on the evidences on the parent nodes, we perform a probabilistic inference to calculate the posterior distribution on the child node. This way, we obtain the detection probability of the given norm in the state indicated by the evidences.

Formally, a Bayesian network $(\mathcal{W}, \mathcal{P})$ expressing a detection model consists of a directed acyclic graph $\mathcal{W} = (\mathcal{F}' \cup \{\textsc{Detect}\}, \mathcal{L})$ and a set of conditional probability distributions $\mathcal{P}$. In the graph $\mathcal{W}$, the set $\mathcal{F}' \subseteq \mathcal{F}$ contains the parent nodes of the network, and each $f_i \in \mathcal{F}'$ represents a feature of the state space. Detect is the child node that provides the detection probabilities. The set $\mathcal{L}$ contains the directed arcs from the parent nodes to the child node. The elements of $\mathcal{L}$ are tuples $(f_i, \textsc{Detect})$ that specify causal relation between $f_i \in \mathcal{F}'$ and Detect. Each feature node $f_i \in \mathcal{F}'$ has a finite set of mutually exclusive values and an unconditional probability distribution $p_i \in \mathcal{P}$. The set $\mathcal{P}$ also contains the conditional probability distribution of Detect, which is conditionally dependent on the features represented by the parent nodes in $\mathcal{F}'$.

Figure 3.1 shows an example of Bayesian network which expresses the detection model for the norm shown in Example 4. In this network, the detection probabilities are conditionally dependent only on the features Street and Cell. Notice that the values of these features have been grouped in order to produce an even more compact representation.



	{00,03,04,07}		{01,02,05,06}	
	{00...04,19...23}	{05...18}	{00...04,19...23}	{05...18}
Yes	0.01	0.10	0.05	0.40
No	0.99	0.90	0.95	0.60

Figure 3.1: Compact detection model for the norm in Example 4.

Using this compact representation of the detection model, the computation of $\mathcal{D}(q, s)$ is made as follows: (i) select the network that represents the detection model of the norm q (a single network can express the detection model of multiple norms), (ii) use the state s to set the evidences on the parent nodes of the network, and (iii) perform a causal reasoning to calculate the detection probabilities of q in s.

Figure 3.2 illustrates the outcome of two probabilistic inferences over the Bayesian network specified in Figure 3.1 using different evidences (see label e in the networks). In Figure 3.2 (a), the evidences indicate that the agent is in a peripheral region, which has a low detection probability:

$$\mathsf{Pro}(\textsc{Detect}{=}\textsc{Yes} \mid \textsc{Street} \in \{00, 03, 04, 07\}, \textsc{Cell} \in \{00...04, 19...23\}) = 0.01;$$

while in Figure 3.2 (b), the evidences indicate that the agent is in a central region, which has a high detection probability of wrong-way driving:

$$\mathsf{Pro}(\textsc{Detect}{=}\textsc{Yes} \mid \textsc{Street} \in \{01, 02, 05, 06\}, \textsc{Cell} \in \{05...18\}) = 0.40.$$

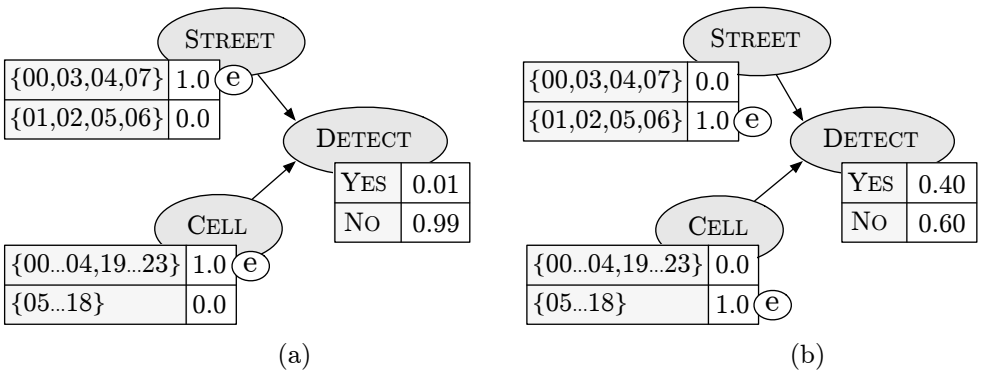


Figure 3.2: Example of outcomes of an inference process over the Bayesian network shown in Figure 3.1.

3.4 Discussion

This chapter has introduced a mathematical formal framework for modeling normative sequential decision problems in stochastic domains. This framework, namely NMDP, generalizes the well-known MDP framework by including explicit representations of norms and detection probabilities of norm violations, and methods for identifying

conflicts between norms and between sanctions. This generalization provides a framework upon which we can develop general algorithms for normative sequential decision making in rational agent models, such as the norm-compliant and self-interested agent models put forward in Chapter 4.

From a theoretical viewpoint, many of the normative languages shown in Section 2.3 provide principles that allow their implementation in a wide range of normative multiagent systems. From a practical perspective, as commented in Section 2.6, there is a gap between the current normative languages and the efforts made in the MDP community to coordinate and improve the tractability of the processes. To the best of our knowledge, the extension put forward in this chapter is the first attempt to bridge this gap [45, 46, 47]. In Oh et al. [87], norms are explicitly represented and used with MDPs, however, no generalization of the MDP framework is formalized. Furthermore, neither sanctions nor probabilities of detecting norm violations are considered, which limits the range of applications of their approach.

Chapter 4

Norm-aware sequential decision making agents

There are no morals about technology at all. Technology expands our ways of thinking about things, expands our ways of doing things. If we're bad people we use technology for bad purposes and if we're good people we use it for good purposes.

Herbert Simon

This chapter puts forward two norm-aware agent models, namely *self-interested* and *norm-compliant*, capable of normative reasoning. Both agent models use the NMDP framework as knowledge representation, however, they reason about norms in distinct ways: the self-interested agent prioritizes the maximization of utilities over the compliance with norms, while norm-compliant agent prioritizes the norm-abiding behaviour over the utility maximization. These distinct normative reasonings are based on distinct notions of rationality, which highlights the relativity of this concept. The self-interested agents are capable of thinking through all possible outcomes and choosing the actions that bring about the best possible outcome, which is *perfect* or

economic rationality[1]. The norm-compliant agents, on the other side, choose that course of action which results in the best possible norm-compliant outcome. I refer to this as *economic norm-compliant rationality*, which can be seen as a type of *bounded rationality*[2] given that the norm-compliant agents do not think through all possible outcomes, only through those that comply with the norms.

In Cognitive and Social Action [33, p. 90], Conte and Castelfranchi state:

> Normative reasoning is neither necessary nor sufficient for a norm-abiding behaviour to occur: (a) *it is by no means necessary* since it might occur either accidentally or out of simple imitation; (b) *it is not sufficient* since normative reasoning and decision-making may produce a transgression of norms, ... Therefore, to model the normative reasoning does not imply modeling a norm-abiding system.

The normative reasoning performed by norm-compliant and self-interested agents *is not necessary* for a norm-abiding behaviour. Even ignoring the existence of norms ruling the system, these agents may accidentally choose actions that respect them. In the self-interested model, the normative reasoning *is not sufficient* for a norm-abiding behaviour, given that the agents implemented according to this model violate the norms if the expected utility of doing so exceeds the expected utility of following them. On the other side, in the norm-compliant model, the normative reasoning *is sufficient* for a norm-abiding behaviour, that is, the decisions taken by the norm-compliant agents introduced in this chapter never produce transgressions of norms.

This chapter is organized as follows. Section 4.1 presents elements and assumptions shared by the agent models proposed in this chapter. Section 4.2 introduces the self-interested agent model, and Section 4.3 introduces the norm-compliant agent model. Finally, Section 4.4 closes the chapter by promoting a discussion about these norm-aware agent models.

[1] In economics and game theory, the agents are sometimes considered to have perfect or economic rationality, that is, they are perfectly logical and geared toward maximum economic gain. This is the case of the self-interested agent model, presented in Section 4.2, which explores its entire state space in order to compute an optimal policy that maximizes its expected utilities.

[2] Concept introduced by Herbert Simon [109] to define that decision-makers in real-world situations have to face constraints such as limited, often unreliable, information regarding the possible alternatives and their consequences, and limited capacity to process the information that is available. The norm-compliant agents, presented in Section 4.3, are boundedly rational given that they always comply with the norms and do not exploit the entire state space as the self-interested agents do.

4.1 Common elements of the agent models

Generally speaking, the problem addressed by the agent models proposed in this thesis is how to reason about norms with the NMDP framework in such a way that we can quantitatively assess the impact of a particular set of norms on the performance of the agents. Although each agent model addresses this problem from different perspectives, both models share the same structure as we can see in Figure 4.1.

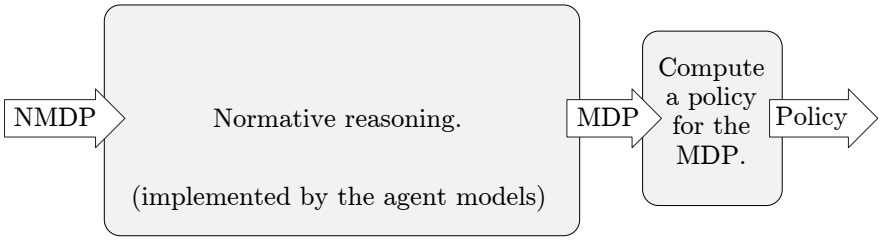


Figure 4.1: Control flow of the agent models, consisting of a normative reasoning over the NMDP input, followed by the policy computation for the resulting MDP.

The following delineate the control flow of the agent shown in Figure 4.1:

- *Agent knowledge*: The knowledge input into the agents is an NMDP, which represents the environment where the agents are launched. It is assumed that this environment is stochastic, memoryless[3] and ontologically fixed (properties captured by the NMDP framework). The agents assume that the NMDP instance is an accurate and complete representation of the environment, even though this is not the case.

- *Normative reasoning*: This process transforms an NMDP input into an MDP that can be computationally solved with known algorithms. The way the normative reasoning is performed depends on the agent model: while the self-interested individuals are norm-autonomous, the norm-compliant ones are not. Section 4.2 and Section 4.3 details the normative reasoning of the self-interested and norm-compliant agents, respectively.

[3] A stochastic process is memoryless, or has the *Markov property*, if the conditional probability distribution of future states of the process depends only upon the present state, not on the sequence of events that preceded it.

- *Policy computation*: The quantitative assessment of the impact of a particular set of norms on the performance of an agent is made on the basis of the expected utility of the agent's policies. To compute the policies (solve the MDP) we can employ well-known algorithms, such as Value Iteration [6], Policy Iteration [61], or Modified Policy Iteration [115, 95].

4.2 Self-interested agent model

A self-interested agent violates a norm only if the expected utility of doing so exceeds the expected utility of complying with the norm. The control flow of this agent model, shown in Figure 4.2, consists of identifying which states violate which norms, and then, representing the respective sanctions within these states. Finally, the agent constructs a policy for the resulting MDP.

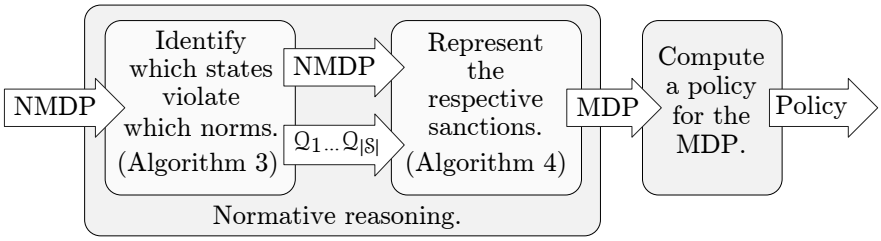


Figure 4.2: The control flow of the self-interested agent model consists of three serial processes: identification of which states violate which norms (Algorithm 3), representation of sanctions (Algorithm 4) and policy computation.

Embodying norms in a normative model in the form of hard-wired constraints [111] has been remarked as problematic for many applications given that it automatically enforces the norms and constrains the ability of mirroring changes in the set of norms [28]. Bearing in mind this fact, the algorithms specified in this chapter have declarative descriptions of norms as input.

Algorithm 3 specifies how to identify which norms are violated in which states. During the algorithm's execution, the norms violated in $s \in \mathcal{S}$ are added to $\mathcal{Q}_s$. For all norms $q \in \mathcal{N}$ (L3), the algorithm checks if the agent is an addressee of q (L4): if so, it assesses $\mathcal{S}_q^{\nabla}$, the set of states that violate q (L5–L7). For all states in $\mathcal{S}_q^{\nabla}$, if the violation of q can be detected in s (L9), q is added to $\mathcal{Q}_s$. Finally, the algorithm returns the NMDP with no updates and the sets $\mathcal{Q}_1 \dots \mathcal{Q}_{|\mathcal{S}|}$, where $\mathcal{Q}_i$ contains the norms violated in the i^{th} state in $\mathcal{S}$.

Algorithm 3 – Identify which states violate which norms.

Input: $\langle \mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{T}, \mathcal{R}, \mathcal{N}, \mathcal{D} \rangle$ (NMDP).
Output: $\langle \mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{T}, \mathcal{R}, \mathcal{N}, \mathcal{D} \rangle$ (NMDP),
$\mathcal{Q}_1 \dots \mathcal{Q}_{|\mathcal{S}|}$ (Norms violated in each state).

1. **for all** $(s \in \mathcal{S})$ **do**
2. $\quad \mathcal{Q}_s \leftarrow \emptyset$
3. **for all** $(q = \langle \alpha, \delta, \mathcal{G}, \mathcal{X}, \mathcal{E}, \sigma \rangle \in \mathcal{N})$ **do**
4. $\quad$ **if** $(agent \in \mathcal{G})$ **then**
5. $\quad\quad$ **if** $(\delta = prohibition)$ **then**
6. $\quad\quad\quad \mathcal{S}_q^{\nabla} \leftarrow \mathcal{E}$
7. $\quad\quad$ **else** $\mathcal{S}_q^{\nabla} \leftarrow \mathcal{X} \setminus \mathcal{E}$
8. $\quad\quad$ **for all** $(s \in \mathcal{S}_q^{\nabla})$ **do**
9. $\quad\quad\quad$ **if** $(\mathcal{D}(q, s) > 0)$ **then**
10. $\quad\quad\quad\quad \mathcal{Q}_s \leftarrow \mathcal{Q}_s \cup \{q\}$
11. **return** $\langle \mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{T}, \mathcal{R}, \mathcal{N}, \mathcal{D} \rangle$, $\mathcal{Q}_1 \dots \mathcal{Q}_{|\mathcal{S}|}$

At the moment that the violating states have been identified by Algorithm 3, the respective sanctions are represented within these violating states. That is, any NMDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{T}, \mathcal{R}, \mathcal{N}, \mathcal{D} \rangle$ can be transformed into an regular MDP $\langle \mathcal{S}', \mathcal{A}', \mathcal{C}', \mathcal{T}', \mathcal{R}' \rangle$ that encodes the sanctions.

It is assumed that the sanctions are imposed by an enforcer with no knowledge about the agents' capabilities and policies. This means that norm violations cannot be foreseen by the enforcer. It is also assumed that norm violations are detectable only in the states they take place, and the norm enforcement cannot be made in a later time step when the agent has left the states that violate the norm.

The conflicts between sanctions are dealt with on the basis of the priority of the sanctions (here, the sanction's priority is the norm's priority α). For instance, if $\langle \rho_1, \phi_1 \rangle$ has priority over $\langle \rho_2, \phi_2 \rangle$, ϕ_1 and ϕ_2 must be called in such an order that the first function overrides, possibly partially, the updates made by the second.

The combinations of sanctions in a given state s are provided by the powerset of $\mathcal{Q}_s$ denoted as $\mathsf{Pow}(\mathcal{Q}_s)$. For instance, if $\mathcal{Q}_s = \{q_1, q_2\}$ then $\mathsf{Pow}(\mathcal{Q}_s) = \{\emptyset, \{q_1\}, \{q_2\}, \{q_1, q_2\}\}$.

The outcome state resulting from a combination of sanctions $\mathcal{B} \in \mathsf{Pow}(\mathcal{Q}_s)$ in the state $s \in \mathcal{S}$ is calculated by the function $\mathsf{Out}(s, \mathcal{B})$ as follows:

$$\mathsf{Out}(s, \mathcal{B}) = \begin{cases} \mathsf{Hi}(\mathcal{B}).\sigma.\phi(\mathsf{Out}(s, \mathcal{B} \setminus \mathsf{Hi}(\mathcal{B}))) & if \quad |\mathcal{B}| > 1 \\ \mathsf{Hi}(\mathcal{B}).\sigma.\phi(s) & if \quad |\mathcal{B}| = 1 \end{cases}$$

where $\mathsf{Hi}(\mathcal{B})$ returns the norm in $\mathcal{B}$ with highest priority:

$$\mathsf{Hi}(\mathcal{B}) = \underset{q \in \mathcal{B}}{\operatorname{argmin}} \, (q.\alpha),$$

and $\mathsf{Hi}(\mathcal{B}).\sigma.\phi$ refers to the function ϕ of the sanction σ of the norm $\mathsf{Hi}(\mathcal{B})$. For details on the priority of norms and normative structure, see Definition 5.

The set containing the combinations of sanctions $\mathsf{W}(s, s') \subseteq \mathsf{Pow}(\mathcal{Q}_s)$ which executed in $s \in \mathcal{S}$ bring about $s' \in \mathcal{S}$ is:

$$\mathsf{W}(s, s') = \{\mathcal{B} \in \mathsf{Pow}(\mathcal{Q}_s) \mid (\mathcal{B} \neq \emptyset) \wedge \mathsf{Out}(s, \mathcal{B}) = s'\}.$$

Using the detection probability of the violation of individual norms provided by the function $\mathcal{D}$, the probability that a combination $\mathcal{B} \in \mathsf{Pow}(\mathcal{Q}_s)$ occurs in $s \in \mathcal{S}$ is calculated as follows:

$$\mathsf{Pro}(\mathcal{B}, s) = \prod_{q \in \mathcal{B}} \mathcal{D}(q, s) \prod_{q \in \mathcal{Q}_s \setminus \mathcal{B}} (1 - \mathcal{D}(q, s)).$$

Thus, $\mathcal{T}'(s, a, s')$ is determined as follows:

$$\mathcal{T}'(s, a, s') = \mathcal{T}(s, a, s') \, \mathsf{Pro}(\emptyset, s) + \sum_{\mathcal{B} \in \mathsf{W}(s, s')} \mathsf{Pro}(\mathcal{B}, s) \tag{4.1}$$

The penalty of a combination of sanctions $\mathcal{B} \in \mathsf{Pow}(\mathcal{Q}_s)$ is the sum of the penalties of each sanction in $\mathcal{B}$:

$$\mathsf{Pen}(\mathcal{B}, s) = \sum_{q \in \mathcal{B}} q.\sigma.\rho(s).$$

And $\mathcal{R}'(s, a, s')$ is determined as follows:

$$\mathcal{R}'(s,a,s') = \frac{\mathsf{Pro}(\emptyset, s)\,\mathcal{T}(s,a,s')\,\mathcal{R}(s,a,s') + \sum_{\mathcal{B}\in\mathsf{W}(s,s')} \mathsf{Pro}(\mathcal{B}, s)\mathsf{Pen}(\mathcal{B}, s)}{\mathcal{T}'(s,a,s')} \tag{4.2}$$

Example 7. *Assume an obligation indicating that the direction of the agent* 01 *in the street* 00 *must be right, and a prohibition indicating that it cannot drive with the seatbelt unfastened. Suppose that in a given state* $s \in \mathcal{S}$ *both norms are violated by the agent* 01*: it is driving left in the street* 00 *and its seatbelt is unfastened. Let these norms be respectively named as* q_1 *and* q_2*, then* $\mathcal{Q}_s = \{q_1, q_2\}$*, and let the detection probability of their violations in* s *be* $\mathcal{D}(q_1, s) = 0.7$ *and* $\mathcal{D}(q_2, s) = 0.2$*.*

$$\begin{aligned} q_1 = \langle 1,\ &\textsc{Obligation},\ \{\textsc{Agent}\ 01\},\\ &(\textsc{Street}{=}00),\ (\textsc{Direction}{=}\textsc{Right}),\\ &\langle\ \{\ \top \to -1.0\},\\ &\ \ \{\ \top \to \{(\textsc{Direction}{=}\textsc{Right})\}\ \}\ \rangle\rangle \end{aligned}$$

$$\begin{aligned} q_2 = \langle 23,\ &\textsc{Prohibition},\ \{\textsc{Agent}\ 01\},\\ &(\textsc{Status}{=}\textsc{Moving}),\ (\textsc{Seatbelt}{=}\textsc{Unfastened}),\\ &\langle\ \{\ \top \to -0.5\},\\ &\ \ \{\ \top \to \{(\textsc{Seatbelt}{=}\textsc{Fastened})\}\ \}\ \rangle\rangle \end{aligned}$$

The norms introduced in Example 7 originate four possible combinations of sanctions. Each line in Table 4.1 specifies one of these combinations and its probability of occurrence calculated with the detection probabilities given in the example.

Table 4.1: Probabilities of combinations of sanctions with respect to the norms introduced in Example 7.

$\mathcal{B}$	$\mathsf{Pro}(\mathcal{B}, s)$
$\emptyset$	(1 – 0.7) * (1 – 0.2) = 0.24
$\{q_1\}$	0.7 * (1 – 0.2) = 0.56
$\{q_2\}$	(1 – 0.7) * 0.2 = 0.06
$\{q_1, q_2\}$	0.7 * 0.2 = 0.14

For each combination $\mathcal{B}$ in Table 4.2, $\mathsf{Pen}(\mathcal{B}, s)$ indicates the sum of penalties, and $\mathsf{Out}(\mathcal{B}, s)$ describes the outcome state. For the sake of simplicity, this last column specifies only the updated features. Notice that there are no values for $\mathcal{B} = \emptyset$ given that there is no penalty and no enforced transition when no sanction is imposed.

Table 4.2: Penalties and outcome states of combinations of sanctions with respect to the norms introduced in Example 7.

$\mathcal{B}$	$\mathsf{Pen}(\mathcal{B}, s)$	$\mathsf{Out}(\mathcal{B}, s)$
$\emptyset$	–	–
$\{q_1\}$	–1.0 + 0.0 = –1.0	... (DIRECTION=RIGHT)
$\{q_2\}$	0.0 – 0.5 = –0.5	... (SEATBELT=FASTENED)
$\{q_1, q_2\}$	–1.0 – 0.5 = –1.5	... (DIRECTION=RIGHT) (SEATBELT=FASTENED)

Algorithm 4 specifies how sanctions are represented within the violating states indicated by Algorithm 3. First, from L1 to L6, Algorithm 4 creates the parameters of the MDP to be outputted using the NMDP parameters. The sets $\mathcal{S}'$ and $\mathcal{A}'$ are augmented by one element each, the absorbing state s_ι and the action a_ι, which are used to represent sanctions within the absorbing states which violate some norm. For each $s \in \mathcal{S}$, if there is some sanction to be represented in s (L8) and s is not absorbing (L9), $\mathcal{T}'$ and $\mathcal{R}'$ are determined as specified in the expressions (4.1) and (4.2). Otherwise, if s is absorbing, the algorithm adds a deterministic non-rewarded transition from s to s_ι with the action a_ι (L15–L17). Then, it determines $\mathcal{T}'$ and $\mathcal{R}'$ as specified in the expressions (4.1) and (4.2) (L18–L20).

A self-interested agent searches for a policy that maximizes its expected utilities, no matter the type of policy. In order to construct a policy for the MDP outputted by Algorithm 4, this agent can use any standard method. Regarding the complexity of the algorithms specified in this section, the order of growth of Algorithm 3 is $O(|\mathcal{N}||\mathcal{S}|)$. The time required for Algorithm 4 is $O(|\mathcal{S}|^2|\mathcal{A}|2^{\Psi})$ where $\Psi = max_{s \in \mathcal{S}} |\Omega_s|$ is the largest number of norms that can be violated in any state. Thus, as long as the number of norms that can be violated in a given state is small, it is computationally feasible for a self-interested agent to use the conversion procedure provided and then solve the resultant MDP in order to determine its best policy.

Algorithm 4 – Represent sanctions.

Input: $\langle \mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{T}, \mathcal{R}, \mathcal{N}, \mathcal{D} \rangle$ (NMDP),
$\mathcal{Q}_1 \ldots \mathcal{Q}_{|\mathcal{S}|}$ (Norms violated in each state).
Output: $\langle \mathcal{S}', \mathcal{A}', \mathcal{C}', \mathcal{T}', \mathcal{R}' \rangle$ (MDP).

1. $\mathcal{S}' \leftarrow \mathcal{S} \leftarrow \mathcal{S} \cup \{s_\iota\}$
2. $\mathcal{A}' \leftarrow \mathcal{A} \leftarrow \mathcal{A} \cup \{a_\iota\}$
3. $\mathcal{C}' \leftarrow \mathcal{C}$
4. $\mathcal{T}' \leftarrow \mathcal{T}$
5. $\mathcal{R}' \leftarrow \mathcal{R}$
6. $\mathcal{C}(s_\iota) \leftarrow \emptyset$
7. **for all** $(s \in \mathcal{S})$ **do**
8. **if** $(\mathcal{Q}_s \neq \emptyset)$ **then**
9. **if** $(\mathcal{C}(s) \neq \emptyset)$ **then**
10. **for all** $(a \in \mathcal{C}(s))$ **do**
11. **for all** $(s' \in \mathcal{S})$ **do**
12. *determine* $\mathcal{T}'(s, a, s')$
13. *determine* $\mathcal{R}'(s, a, s')$
14. **else**
15. $\mathcal{C}'(s) \leftarrow \mathcal{C}(s) \leftarrow \{a_\iota\}$
16. $\mathcal{T}(s, a_\iota, s_\iota) \leftarrow 1.0$
17. $\mathcal{R}(s, a_\iota, s_\iota) \leftarrow 0.0$
18. **for all** $(s' \in \mathcal{S})$ **do**
19. *determine* $\mathcal{T}'(s, a_\iota, s')$
20. *determine* $\mathcal{R}'(s, a_\iota, s')$
21. **return** $\langle \mathcal{S}', \mathcal{A}', \mathcal{C}', \mathcal{T}', \mathcal{R}' \rangle$

4.3 Norm-compliant agent model

Starting from a norm-compliant state does not ensure norm-compliance for the agent's whole life. If an agent starts in such a state we can only infer that it is fulfils the norms in this state, but eventually, executing a given policy, this agent may end up in some violating state. If this agent aims at guaranteeing norm-compliance during its whole life, then it must ensure that no state in $\mathcal{S}^\nabla$, the set of states that violate some norm, can be reached. This is exactly what the norm-compliant agent model does.

The control flow of the norm-compliant agent model, shown in Figure 4.3, begins by assessing $\mathcal{S}^\nabla$. Once the assessment of this set has been accomplished, $\mathcal{N}$ and $\mathcal{D}$ are discarded from the NMDP so as to get an regular MDP. Using the norm violating states in $\mathcal{S}^\nabla$, the agent constructs a certainly norm-compliant MDP by confining its state space to $\mathcal{S} \setminus \mathcal{S}^\nabla$ and by restricting its capability function as follows:

$$\mathcal{C}^{\blacktriangledown}(s) = \{a \in \mathcal{C}(s) \mid \mathsf{Imo}(s,a) \cap \mathcal{S}^{\triangledown} = \emptyset\}$$

where $\mathsf{Imo}(s,a)$ gives the possible immediate outcome states of executing the action a in the state s:

$$\mathsf{Imo}(s,a) = \{s' \in \mathcal{S} \mid \mathcal{T}(s,a,s') > 0\}.$$

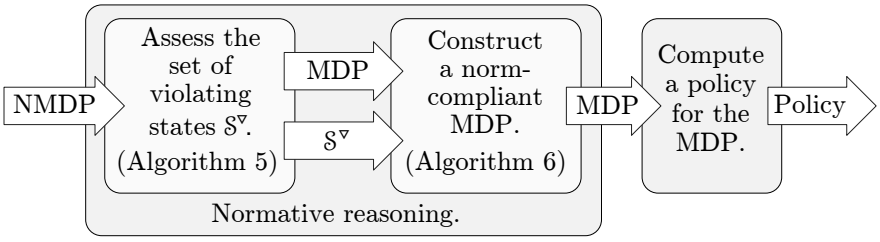


Figure 4.3: The control flow of a norm-compliant agent consists of three serial processes: assessment of $\mathcal{S}^{\triangledown}$ (Algorithm 5), construction of a norm-compliant MDP (Algorithm 6), and policy computation.

Algorithm 5 assesses the norm violating states and stores them in the set $\mathcal{S}^{\triangledown}$. For every norm in $\mathcal{N}$ (L2), the algorithm verifies if the agent is an addressee of this norm. If the norm is a prohibition, the prohibited states are added to $\mathcal{S}^{\triangledown}$ (L4–L5). Otherwise, if the norm is an obligation, then all states in context where the norm applies, except the obliged ones, are added to $\mathcal{S}^{\triangledown}$ (L6). Finally, the algorithm returns $\mathcal{S}^{\triangledown}$ and an regular MDP obtained by eliminating $\mathcal{N}$ and $\mathcal{D}$ from the input NMDP.

Algorithm 5 – Assess $\mathcal{S}^{\triangledown}$.

Input: $\langle \mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{T}, \mathcal{R}, \mathcal{N}, \mathcal{D} \rangle$ (NMDP).
Output: $\langle \mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{T}, \mathcal{R} \rangle$ (MDP),
$\mathcal{S}^{\triangledown}$ (States that violate some norm).

1. $\mathcal{S}^{\triangledown} \leftarrow \emptyset$
2. **for all** $(q = \langle \alpha, \delta, \mathcal{G}, \mathcal{X}, \mathcal{E}, \sigma \rangle \in \mathcal{N})$ **do**
3. **if** $(agent \in \mathcal{G})$ **then**
4. **if** $(\delta = prohibition)$ **then**
5. $\mathcal{S}^{\triangledown} \leftarrow \mathcal{S}^{\triangledown} \cup \mathcal{E}$
6. **else** $\mathcal{S}^{\triangledown} \leftarrow \mathcal{S}^{\triangledown} \cup (\mathcal{X} \setminus \mathcal{E})$
7. **return** $\langle \mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{T}, \mathcal{R} \rangle, \mathcal{S}^{\triangledown}$

Once Algorithm 5 has determined $\mathcal{S}^{\triangledown}$, Algorithm 6 is called to generate a certainly norm-compliant MDP. If there is some violating state (L3), Algorithm 6 limits the capability function $\mathcal{C}^{\blacktriangledown}$ in such a way that no state in $\mathcal{S}^{\triangledown}$ can be reached from states in $\mathcal{S}^{\blacktriangledown} = \mathcal{S} \setminus \mathcal{S}^{\triangledown}$ (L4–L7). This guarantees that the norms are followed if the agent is launched from some state in $\mathcal{S}^{\blacktriangledown}$.

Algorithm 6 – Construct a certainly norm-compliant MDP.

Input: $\langle \mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{T}, \mathcal{R} \rangle$ (MDP),
$\mathcal{S}^{\triangledown}$ (States that violate some norm).
Output: $\langle \mathcal{S}^{\blacktriangledown}, \mathcal{A}, \mathcal{C}^{\blacktriangledown}, \mathcal{T}, \mathcal{R} \rangle$ (MDP).

1. $\mathcal{S}^{\blacktriangledown} \leftarrow \mathcal{S} \setminus \mathcal{S}^{\triangledown}$
2. $\mathcal{C}^{\blacktriangledown} \leftarrow \mathcal{C}$
3. **if** $(\mathcal{S}^{\triangledown} \neq \emptyset)$ **then**
4. **for all** $(s \in \mathcal{S}^{\blacktriangledown})$ **do**
5. **for all** $(a \in \mathcal{C}^{\blacktriangledown}(s))$ **do**
6. **if** $(\mathsf{lmo}(s, a) \cap \mathcal{S}^{\triangledown} \neq \emptyset)$ **then**
7. $\mathcal{C}^{\blacktriangledown}(s) \leftarrow \mathcal{C}^{\blacktriangledown}(s) \setminus \{a\}$
8. **return** $\langle \mathcal{S}^{\blacktriangledown}, \mathcal{A}, \mathcal{C}^{\blacktriangledown}, \mathcal{T}, \mathcal{R} \rangle$

Regarding the complexity of the algorithms specified in this section, the order of growth of Algorithm 5 is $O(|\mathcal{N}||\mathcal{S}|)$, therefore, polynomial in the number of norms and number of states. Algorithm 6 is performed in $O(|\mathcal{S}^{\blacktriangledown}||\mathcal{S}||\mathcal{A}|)$, that is, polynomial in the number of violating states, number of states and number of actions. However, in practice, Algorithm 6 tends to perform better as the capability function limits the number of admissible actions (L5). In these cases, the order of growth in the number of actions is smaller than $|\mathcal{A}|$.

4.4 Discussion

This chapter describes rigorous models of how norms impact the sequential decision making of norm-aware agents. In particular, it describes computational models of normative reasoning for two agent types: *self-interested* agents incorporate sanctions into the state-transition and reward functions, while *norm-compliant* agents use norms to create smaller norm-compliant MDPs. The normative reasoning processes built in these agent models are specified by means of general algorithms which operate on the NMDP framework introduced in Chapter 3.

The norm-compliant agent model covers all the properties, except norm-autonomy as it never deviates from the norms. The main advantage of this model is that it uses norms to prune the decision space, and this way it can ensure coordination with other norm-compliant agents and achieve computational leverage in the policy construction (this is demonstrated in the experiments performed in Chapter 5).

The self-interested agent model covers all the properties analyzed in Section 2.5. To do so, the NMDP framework provides explicit representations of norms and sanctions, while the algorithms built on the top of the framework ensure norm-autonomy by way of quantitative decisions and probabilistic planning. That is, the self-interested agent model provides a quantitative principled method for norm-autonomous decision making on the basis of evaluations of explicitly represented norms and sanctions.

Unlike the MDP extensions for collaborative multiagent systems [17, 9, 59, 5, 54, 7, 83] shown in Section 2.1.4, the approach presented in this thesis allows coordinating self-interested agents. In this case, coordination can be enforced through the imposition of adequate sanctions (norm enforcement intensities, penalty values) which make the self-interested agents to give up on norm violations. Chapter 5 studies the impact of different sanctions in the agents' behavior.

Part III

Experimentation

Chapter 5

Motion problem

It doesn't matter how beautiful your theory is, it doesn't matter how smart you are. If it doesn't agree with experiment, it's wrong.

Richard Feynman

The norm representation and normative reasoning methods for NMDPs give us the chance to *measure* the impact of norms on the sequential decision-making of norm-aware agents, allowing a sort of *quantitative* evaluations of the agents' performance. This chapter provides examples of the type of evaluations that can be made. For this purpose, several experiments with norm-aware agents were performed in a simulated environment that models a motion problem governed by norms.

Section 5.1 presents a general description of the motion problem, which elucidates the motion environment, the actions available to the agents for solving the motion problem, the reward-based performance measurement used to evaluate the quality of the solutions, the coordination problems that may take place in multiagent scenarios, and the normative approach to cope with these problems. Section 5.2 describes how the motion problem, more specifically the motion environment, is represented with the NMDP framework, focusing on the technical details of the NMDP specification. Section 5.3 presents the motion simulator developed to investigate the behaviour of the norm-aware agents in the normative motion environment under study. Finally, Section 5.4 explains the experiments, and Section 5.5 presents and discusses the results of the experiments.

5.1 General description

5.1.1 Motion Environment

The motion environment resembles an urban road network, however, it is not an attempt to provide a faithful representation of a real traffic situation. This environment, shown in Figure 5.1, is made up of discrete cells, where the agents (represented by triangles indicating their direction) are able to move one cell at a time. There are 8 streets, 4 vertical and 4 horizontal (16 intersections), and each street contains 24 contiguous cells. There are 8 gateways via which the agents enter and leave the environment: north-west (NW), north-east (NE), south-west (SW), south-east (SE), west-north (WN), west-south (WS), east-north (EN) and east-south (ES). Once placed in a gateway, an agent decides between two adjoining streets. For example, in Figure 5.1 (a), the agent in the gateway WS decides between STREET=02 and STREET=03. Taken this decision, the agent is free to move around using the available actions. Its interaction with the environment ends when it leaves through a gateway. For example, the agent highlighted in Figure 5.1 (b) is terminated when it leaves the environment via the gateway EN.

From an agent's perspective, the *motion problem* consists of moving from an origin gateway, where the agent is launched from, to a destination gateway, where the agent receives a positive reward (incentive). Each agent has its own origin and destination gateways. The motion environment is *partially observable*: the agents always know their own position, but they do not know each other's position. To reach the destination gateways, the following actions are available for the agents:

- *Start*: This action is the only one available in the gateways, and it consists of starting in one of the two adjoining streets. For example, in Figure 5.1 (a) the agent in the gateway WS decides between street 02 and 03. Taken this decision, the agent moves into the next cell of the chosen street.
- *Move*: Consists of moving to an adjacent cell, which is calculated based on the current direction of the agent. If there is no such an adjacent cell, as illustrated in Figure 5.1 (c), the agent stays in the same cell.
- *Turn*: This action changes the agent's direction based on the rotation parameter, clockwise or counterclockwise.

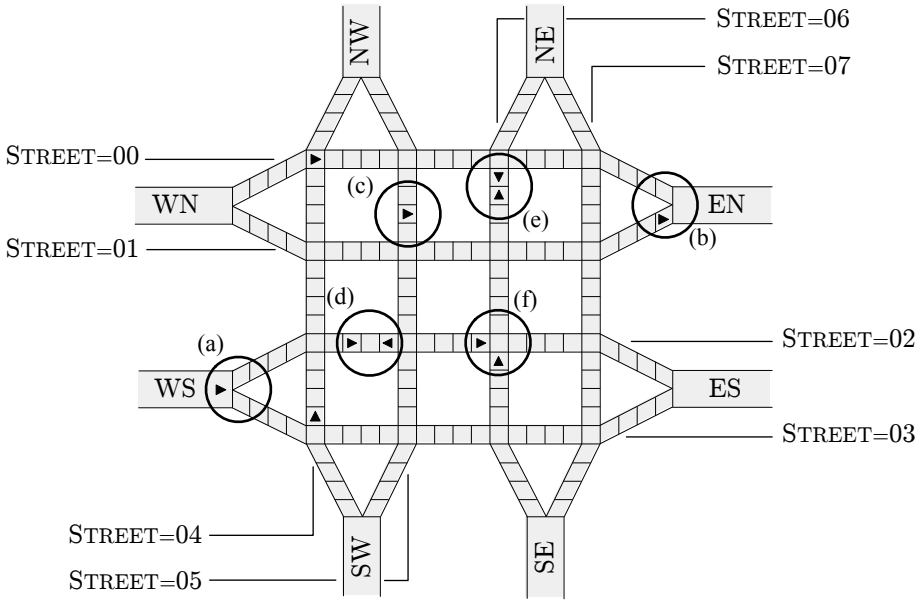


Figure 5.1: Normative multiagent motion environment populated with agents represented by triangles. Situation (a) shows an agent entering the environment through the gateway WS. Situation (b) shows an agent that is about to leave the environment through the gateway EN. In Situation (c), if the agent executes *Move*, it remains in the same cell. Situation (d) and (e) indicate crash situations, and Situation (f) gives an intersection approaching example.

- *Stop*: This action is available only if the agent is moving, and it consists of stopping in an adjacent cell, which is calculated based on the current direction of the agent. Furthermore, if there is no such an adjacent cell, the agent stays in the same cell.

- *End*: This action is available only in the cells adjacent to a gateway, and corresponds to leaving the environment using the adjoining gateway. For example, if the agent shown in Figure 5.1 (b) executes this action, it leaves the environment via the gateway EN.

Non-determinism is modeled by the fact that *Move*, *Turn* and *Stop* are unreliable: their intended outcome occurs with probability 0.99, but with probability 0.01 the agent remains in the same position. The agents' actions are handled synchronous with the motion environment, that is, each agent executes one action per time slice (the synchronization of actions is detailed in Section 5.3 with the motion simulator).

For every state-transition implied by an action, the agent receives a reward, which may be positive (motivation) or negative (cost). For this environment, the reward is $-0.01\mathsf{R_N}$ for all state-transitions ($\mathsf{R_N}$ is the normalizing constant), except for stopping which gives $-0.05\mathsf{R_N}$ and ending in the destination gateway which gives $+0.6\mathsf{R_N}$ (an incentive to reach this gateway quickly). Notice that leaving the environment via any other gateway gives $-0.01\mathsf{R_N}$.

5.1.2 Coordination problem and normative solution

To successfully tackle the motion problem in a multiagent setting, it may not be enough to find a path from the origin gateway to the destination gateway. It is also important to avoid crashes along the route. If an agent crashes, its interaction with the environment is immediately terminated, which makes impossible the achievement of the destination gateway.

A crash occurs when agents coming from different cells try to occupy the same cell or when agents occupying adjacent cells try to cross each other in opposite directions. The first case is illustrated in Figure 5.1 (d), where two agents are about to move into the same cell, and the second case is shown in Figure 5.1 (e), where two agents are about to cross each other. Notice that agents moving to the same direction can move into the same cell with no problems. At intersections, crashes can be avoided if the agents stop before crossing. It is assumed that in this way the agents are slow enough to safely cross the intersection, either autonomously or with the help of some central mechanism provided by the environment infrastructure. The following rule is applied at intersections: if $n > 1$ agents do not stop before moving into an intersection, these n agents crash; any agent that stops before moving into this intersection is immune to crashes. In Figure 5.1 (f), if one of the agents stops before crossing, no one crashes, and if no one stops, both crash.

To cope with the motion problem, more specifically with the coordination problems that cause crashes, I will introduce two groups of norms to regulate the traffic of agents in the motion environment. The first group specifies a flow direction for each street. With these norms, it is expected to avoid crashes such as the ones shown in Figure 5.1 (d) and (e). The second group aims at forcing any agent travelling along horizontal streets to stop before crossing intersections, thus giving priority to agents on vertical streets. With these norms, it is expected to avoid crashes such as the one illustrated in Figure 5.1 (f). Detected violations of these norms are punished with penalties, and

the transgressors are forced behave according to the norms.

In the motion environment, autonomous transgressions of norms are allowed, and the detection and sanctioning of norm violations is carried out by an enforcement mechanism. As this mechanism is not capable of observing and controlling the internal states and processes of the agents, norm enforcement is based on the detection of violating states in terms of observations in the environment. However, this mechanism is assumed to be resource-bounded, so the detections are made only to some extent, that is, some violations may go unpunished. In this context, consider the following assumptions:

(i) If a norm violations is not immediately detected, it is no longer detectable (there is no history of the environment, and the agents do not blow the whistle on any of their acquaintances). If a norm violation is detected, the respective sanction is imposed immediately.

(ii) The detection probability is the same, regardless of the state and the norm that has been violated in this state.

(iii) The agents do not know if their norm violations will be detected, but they always know the detection probabilities[1].

5.2 NMDP instances

This section describes the NMDP instances, which represent the motion environment presented in Section 5.1 from the perspective of the individual agents. First of all, there are 8 NMDP instances, one for each possible destination gateway. The only difference between the instances is on the reward function, which gives the positive reward for ending in the corresponding destination gateway. That is, each instance incites to reach a different destination gateway. All the remaining components, from state space to detection function, are shared by all NMDP instances. Said this, the following details the specification of the NMDP instances (for a detailed specification, see Appendix B).

From the perspective of an agent, the environment can be observed only partially. An agent always knows its current street, cell, direction and status, but has no infor-

[1] Although the assessment of these probabilities is essential to several applications, this problem is outside the scope of this thesis, which assumes that these probabilities have already been assessed and focuses on the normative reasoning problem.

mation about the other agents in the environment. More precisely, the agents do not know that there are other agents running in the same environment. This can be seen in the NMDP instances that represent neither features about other agents nor joint actions. In this context, norms are expected to cope with the coordination problems between these agents, which are incapable of observe each other.

The state space $\mathcal{S}$, specified in Table 5.1, is composed of 4 features: STREET, CELL, DIRECTION and STATUS. The features STREET and CELL designate the location of the agent. There are 8 streets made up of 24 contiguous cells, and there are 8 gateways, treated as streets, composed of a single cell named GATEWAY. The valid combinations of values for STREET and CELL are numbered streets with numbered cells, and non-numbered streets (gateways) with GATEWAY[2]. There are 4 possible directions: RIGHT, LEFT, UP and DOWN. There are 2 possible statuses: STILL if the agent is holding its position; and MOVING if the agent is in motion.

Table 5.1: Factored state space representing the motion environment.

i	f_i	$\mathcal{V}_{f_i}$
1	STREET	{00, 01, ... 07, NW, NE, SW, SE, WN, WS, EN, ES}
2	CELL	{00, 01, ... 23, GATEWAY}
3	DIRECTION	{UP, RIGHT, DOWN, LEFT}
4	STATUS	{STILL, MOVING}

An intersection is made up of two overlaying cells that represent the same location, but belonging to different streets. So, when an agent occupies an intersection, which STREET and which CELL is this agent in? To answer this question, we must check the agent's current direction. If DIRECTION$\in${UP,DOWN}, the agent is in the vertical street, and if DIRECTION$\in${LEFT,RIGHT}, the agent is in the horizontal street. Figure 5.2 shows the four possible combinations of values for STREET, CELL and DIRECTION at a given intersection, where STREET=X meets STREET=X', and CELL=i and CELL=j represent the same location. Figure 5.2 (a) and (b) illustrate situations where DIRECTION$\in${LEFT,RIGHT}, which puts the agent in the horizontal street, while (c) and (d) illustrate situations where DIRECTION$\in${UP,DOWN}, which puts the agent in the vertical street.

[2]The agent's capabilities (admissible actions) are defined in such a way that invalid combinations of STREET and CELL cannot occur.

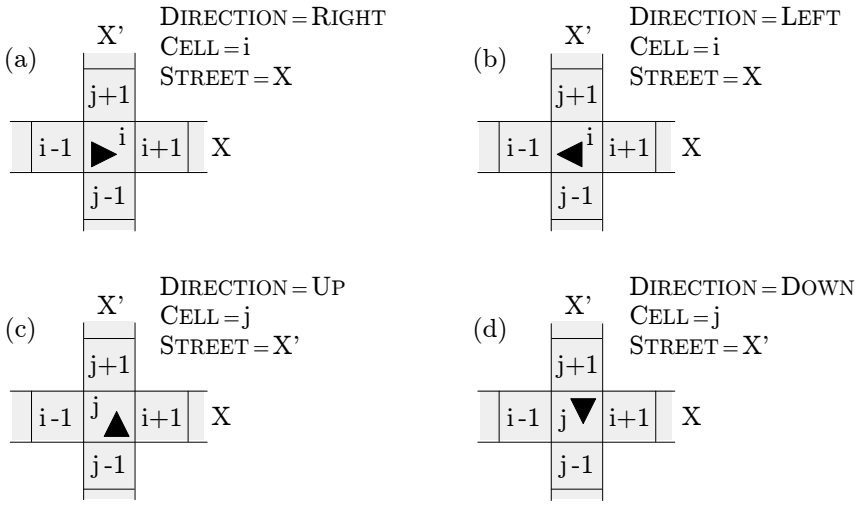


Figure 5.2: Situations (a) and (b): if DIRECTION∈{LEFT,RIGHT} the agent is in the horizontal street X. Situations (c) and (d): if DIRECTION∈{UP,DOWN} the agent is in the vertical street X'. Notice that i is part of X and j is part of X'.

The action space consists of 5 actions,

$$\mathcal{A} = \{\text{START(s)}, \text{MOVE}, \text{TURN(r)}, \text{STOP}, \text{END}\},$$

which are detailed as follows:

- START(s): The agent is capable of executing this action only when STREET∈{NW, NE, SW, SE, WN, WS, EN, ES} and CELL=GATEWAY, that is, the agent is placed in a gateway. The parameter s indicates the street where the agent decided to start in. This action operates over the features STREET and CELL. For instance, if the agent in Figure 5.1 (a) executes START(02), STREET changes from WS to 02, and CELL changes from GATEWAY to 00. The reward for this action is $-0.01\mathsf{R_N}$ from any gateway to any adjacent cell.

- MOVE: This action is available in states where STREET∈{00, 01, ... 07} and CELL∈{00, 01, ... 23}. This action updates the value of CELL based on the current value of DIRECTION, and changes the value of STATUS to MOVING if its current value is STILL. As mentioned in Section 5.1, the intended outcome of this action occurs with probability 0.99, but with probability 0.01 CELL and STATUS stay unchanged. The reward for executing MOVE is $-0.01\mathsf{R_N}$ from any origin state to any outcome state. For example, if an agent executes MOVE in the state

(Street=04, Cell=05, Direction=Up, Status=Moving)

with probability 0.99 the outcome is

(Street=04, Cell=06, Direction=Up, Status=Moving)

and with probability 0.01 it remains in the same state.

- Turn(r): The agent is capable of performing this action in states where Street ∈{00, 01, ... 07} and Cell∈{00, 01, ... 23}. This action updates Direction based on the rotation parameter r∈{Clockwise, Counterclockwise} and changes the value of Status to Moving if its current value is Still. At intersections, turning also updates Street and Cell based on the new Direction (see Figure 5.2). The intended outcome of this action occurs with probability 0.99, but with probability 0.01 all features stay unchanged. The reward for this action is $-0.01R_N$ from any origin state to any outcome state. For example, if an agent executes Turn(Clockwise) in the following intersection

  (Street=00, Cell=04, Direction=Right, Status=Moving)

  with probability 0.99 the outcome is

  (Street=04, Cell=19, Direction=Down, Status=Moving)

  and with probability 0.01 it remains in the same state. Notice that the value of Street is expected to change from 00 to 04 (the perpendicular street), and Cell is expected to change from 04 to 19 (the overlying cell).

- Stop: This action is available in any state where Street∈{00, 01, ... 07}, Cell∈{00, 01, ... 23} and Status=Moving. Like in the action Move, the value of Cell is updated based on the current value of Direction. The difference between these actions is that Stop changes Status from Moving to Still. The intended outcome of this action occurs with probability 0.99, but with probability 0.01 Cell and Status stay unchanged. The reward for this action is $-0.05R_N$ from any origin state to any outcome state. For example, if an agent executes Stop in the state

  (Street=04, Cell=05, Direction=Up, Status=Moving)

with probability 0.99 the outcome is

$$(\textsc{Street}{=}04,\ \textsc{Cell}{=}06,\ \textsc{Direction}{=}\textsc{Up},\ \textsc{Status}{=}\textsc{Still})$$

and with probability 0.01 it remains in the same state.

- End: The agent is capable of ending only in states where Cell$\in\{00, 23\}$, that is, cells adjacent to gateways. This action operates over Street and Cell. For instance, if the agent in Figure 5.1 (b) executes End, Street changes from 01 to EN, and Cell changes from 23 to Gateway. The reward for this action is $+0.6\mathsf{R_N}$ from any state to the destination gateway, and $-0.01\mathsf{R_N}$ from any cell to the remaining gateways. As highlighted in the beginning of this section, here is the only difference between the NMDP instances: each of them rewards $+0.6\mathsf{R_N}$ for ending in a different destination gateway.

As mentioned previously, the agents modeled in this chapter are unaware about the presence of other agents operating in the motion environment, which may give rise to crashes. In this context, norms are expected to cope with the coordination problems between these agents, which are incapable of observe each other. As said in Section 5.1, the norms active in this environment are divided in two groups. The first group, named $\mathcal{N}_1$, is addressed to all agents in the system and specifies a flow direction for each street. If a violation of these norms is detected, the agent is set to the intended direction and loses $\lambda\mathsf{R_N}$. The coefficient λ multiplies $\mathsf{R_N}$ to determine the penalty imposed by the sanction (Section 5.4 specifies the range of values of λ that have been used in the experiments). The norms in $\mathcal{N}_1$ are specified using the following template, where α is the norm priority, $\top$ means true (the norm is addressed to all agents in the system), X is a particular street and Y is its flow direction:

$$\begin{array}{l}\langle \alpha,\ \textsc{Obligation},\ \top,\\ \qquad (\textsc{Street}{=}\mathrm{X}),\\ \qquad (\textsc{Direction}{=}\mathrm{Y}),\\ \qquad\quad \langle\ \{\ \top \rightarrow -\lambda\mathsf{R_N}\ \},\\ \qquad\qquad \{\ \top \rightarrow \{(\textsc{Direction}{=}\mathrm{Y})\}\ \}\ \rangle\rangle\end{array}$$

The second group of norms, named $\mathcal{N}_2$, aims at forcing any agent travelling along horizontal streets to stop before crossing intersections, thus giving priority to agents on vertical streets. If a violation of these norms is detected, the agent loses $\lambda\mathsf{R_N}$ and is forced to stop. The norms in $\mathcal{N}_2$ are specified according to the following template, where α is the priority of the norm, $\top$ means that this norm is addressed to all agents,

X is a horizontal street, Y is the obliged flow direction in this street, and Z is a cell adjacent to an intersection:

$$\langle \alpha, \textsc{Prohibition}, \top, (\textsc{Street}{=}\mathrm{X}) \wedge (\textsc{Direction}{=}\mathrm{Y}) \wedge (\textsc{Cell}{=}\mathrm{Z}), (\textsc{Status}{=}\textsc{Moving}), \langle \{ \top \rightarrow -\lambda \mathsf{R_N} \}, \{ \top \rightarrow \{(\textsc{Status}{=}\textsc{Still})\} \} \rangle \rangle$$

Figure 5.3 indicates with arrows the traffic direction for each street as determined by $\mathcal{N}_1$. This figure also indicates with the label S the cells along horizontal streets in which the agents must stop to give priority to agents travelling in the vertical streets.

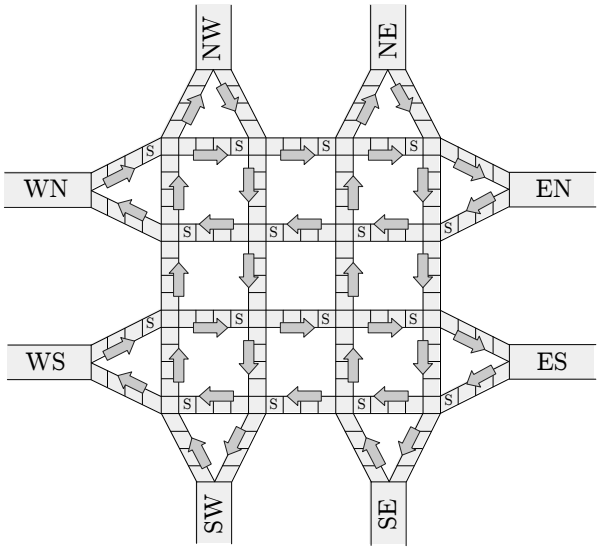


Figure 5.3: Traffic direction for each street, indicated by the arrows, and cells where the agents must stop before moving into the adjacent intersection, indicated by the label S.

The full set of norms ruling the environment, known by all agents, is $\mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2$. The set of norms $\mathcal{N}$ is consistent (see Definition 11), and their respective sanctions are free of conflicts in $\mathcal{S}$ (see Definition 12). The number of norms in $\mathcal{N}$ is proportional to the number of the streets: in $\mathcal{N}_1$ there are 8 norms, each one ruling the traffic flow in the corresponding street, and in $\mathcal{N}_2$ there are 16 norms, each one at the corresponding intersection, ruling the access of agents coming from horizontal streets. As specified in Appendix B, α ranges from 01 to 24, denoting the unique identifier of the norms.

As said in Section 5.1, the detection probability of norm violations is the same, regardless of the state and the norm that has been violated in this state. This makes the specification of $\mathcal{D}$ quite simple, given that $\mathcal{D}(q, s) = \beta$ for any q and s. Section 5.4 specifies the range of values of β used in the experiments. A more sophisticated detection function, which varies with norms and states, will be presented in the case study of Chapter 6.

The following remarks close this section:

- There are 8 NMDP instances, each of them providing an incentive by means of a positive reward for ending in a different destination gateway. These instances, combined with the agent models, will compose the agent classes employed later in Section 5.4 to create the agent populations.
- The range of values of λ (penalty coefficient), as well as the range of values of β (detection probability), are part of the settings utilized in the experiments. Section 5.4 specifies these settings, which allow to run experiments with different situations of the environment.
- Any gateway provides access to two streets with different traffic direction rules, so the agents always have the option of complying with $\mathcal{N}$ since the beginning of their lives, regardless the gateway they are launched from.
- The agents know the topology of the road network and their own location in this network, but they have no sensors to perceive the presence of other agents. This limitation gives rise to coordination problems (crashes) that affect their utilities. In this context, it is expected that the norms help with these problems.

5.3 Simulator

The motion simulator is an implementation of the motion environment described in Section 5.1 programmed with the Java Platform Standard Edition (Java SE)[3].

[3]http://www.oracle.com/us/technologies/java/overview/index.html

In each simulation, the simulator, in a synchronous cyclic fashion, provides sensory information to the participating agents and expects their actions. Time is represented as discrete time slices, and within a time slice the simulator collects the agents' actions, executes the actions, and provides to the agents their respective percepts containing the outcome of their actions. Figure 5.4 provides an overview of a simulation, where an active population of agents receives percepts of the simulator (s_{ai} is the current state of the agent ai) and informs the actions to be executed ($\pi(s_{ai})$ is the action to be executed by the agent ai in the state s_{ai}). The simulator regulates the simulation according to the norms and the detection function provided. To illustrate the motion environment as the simulation runs, the simulator communicates with a graphical interface, where the self-interested agents are depicted as red triangles and the norm-compliant agents are drawn as blue triangles. The triangles' orientation indicates the agents' current direction. The simulator saves the *utilities* gained by the agents, the number of agents that *crash*, and the *penalties* paid by the agents for detected norm violations. With these data, we can calculate the average utility per agent, percentage of agents that crash, and sum of penalties paid by the agents, which makes possible to compare agent models, as well as sets of norms.

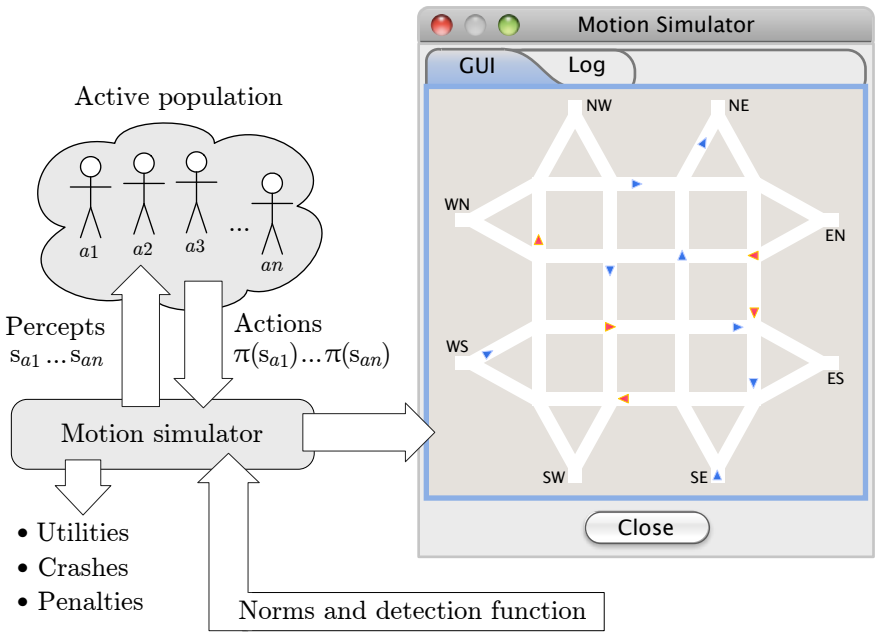


Figure 5.4: Simulation overview.

The simulator, for a specific environment setting, determines the *real utility* that an agent gets from its policy, which not necessarily corresponds to the *expected utility* calculated on the basis of a partial view of the environment (as previously mentioned, the agents cannot observe each other, which in multiagent settings may cause crashes, which in turn, prevent the involved agents from receiving the positive reward on their respective destination gateway). For this reason, the simulator determines the state and the utility (accumulated rewards) of all agents at each time slice based on their actions utilizing: (i) the state-transition probabilities and the rewards given in Section 5.1.1; (ii) the crash rules provided in Section 5.1.2; and (iii) the penalties and detection probabilities (later detailed in Section 5.4 with the experimental settings). Notice that the state-transition probabilities and the reward values are accurately represented by the agents, as we can see in Section 5.2.

Figure 5.5 depicts the interaction protocol between an agent and the simulator, expressed as a sequence diagram. First, the agent requests to join the motion simulator, which has a choice of either accepting or rejecting the proposal (the $\times$ in the decision diamond indicates an exclusive *or* decision). If the agent is accepted, it will inform the simulator about its action (the first action is based on the initial state, which the agent already knows). In the following step, the simulator informs the agent's perception, which is the outcome state resulting from its action. This action-perception interaction (illustrated in Figure 5.4 as well) repeats until the agent crashes or exits the environment through some gateway. At the moment that one of these two events occur, the simulator informs the agent about the termination of their interaction.

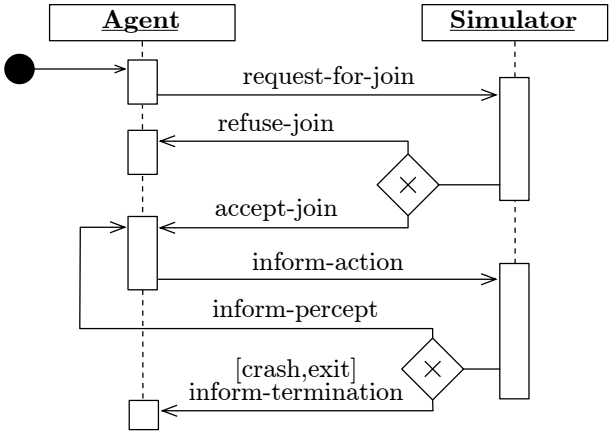


Figure 5.5: Interaction protocol between an agent and the simulator.

The simulation state-transition is as follows:

1. *Handle joining requests from new agents.* A request-for-join message contains a sender (the identification of the agent that intends to join the simulator), a receiver (simulator), an initial state where the agent requests to start in, and the agent's destination gateway, so the simulator knows the gateway where the agent receives the positive reward. For example, consider the following message from Agent01 (moving to the destination gateway SE) to Simulator, requesting to join the environment via the gateway WN:

⟨REQUEST–FOR–JOIN⟩
⟨SENDER⟩ Agent01 ⟨/SENDER⟩
⟨RECEIVER⟩ Simulator ⟨/RECEIVER⟩
⟨INITIAL–STATE⟩
⟨STREET⟩ WN ⟨/STREET⟩
⟨CELL⟩ Gateway ⟨/CELL⟩
⟨DIRECTION⟩ Right ⟨/DIRECTION⟩
⟨STATUS⟩ Still ⟨/STATUS⟩
⟨/INITIAL–STATE⟩
⟨DESTINATION⟩
⟨GATEWAY⟩ SE ⟨/GATEWAY⟩
⟨/DESTINATION⟩
⟨/REQUEST–FOR–JOIN⟩

If the message is understood, the simulator accepts the request and informs the current iteration (time slice) of the simulator, used to synchronize actions:

⟨ACCEPT–JOIN⟩
⟨SENDER⟩ Simulator ⟨/SENDER⟩
⟨RECEIVER⟩ Agent01 ⟨/RECEIVER⟩
⟨ITERATION⟩ 1 ⟨/ITERATION⟩
⟨/ACCEPT–JOIN⟩

Otherwise, the simulator refuses the request for join:

⟨REFUSE–JOIN⟩
⟨SENDER⟩ Simulator ⟨/SENDER⟩
⟨RECEIVER⟩ Agent01 ⟨/RECEIVER⟩
⟨/REFUSE–JOIN⟩

2. *Collect and execute all actions.* In the motion simulator, the actions are executed synchronously, that is, the simulator collects all actions (one per agent) for the current iteration (time slice), and then it executes the actions. As the simulator implements a stochastic normative environment, the outcome of the actions is uncertain and depends on the detection of norms violations. The transition rules of the simulator have been implemented according to the general description given in Section 5.1. The following message gives an example of inform-action, which informs the simulator that Agent01 decided to start in street 00:

   ⟨INFORM–ACTION⟩
   ⟨SENDER⟩ Agent01 ⟨/SENDER⟩
   ⟨RECEIVER⟩ Simulator ⟨/RECEIVER⟩
   ⟨ACTION⟩ Start(00) ⟨/ACTION⟩
   ⟨ITERATION⟩ 1 ⟨/ITERATION⟩
   ⟨/INFORM–ACTION⟩

3. *Inform terminated agents.* All agents that crash or leave the environment receive a message informing that they have been terminated. The inform-termination message also includes the cause of the termination. For instance, the following message informs Agent01 that it has been terminated because it crashed in the iteration 21:

   ⟨INFORM–TERMINATION⟩
   ⟨SENDER⟩ Simulator ⟨/SENDER⟩
   ⟨RECEIVER⟩ Agent01 ⟨/RECEIVER⟩
   ⟨ITERATION⟩ 21 ⟨/ITERATION⟩
   ⟨CAUSE⟩ Crash ⟨/CAUSE⟩
   ⟨/INFORM–TERMINATION⟩

4. *Deliver the percepts.* Each agent still active in the simulator receives a sensory information by indicating its current street, cell, direction and status. This is a self-perception given that any information about the other agents running in the simulator is not provided. For example, after executing the action Start(00) in the iteration 1 (first time slice), Agent01 receives an inform-percept message informing its state in the iteration 2 (second time slice):

```
⟨INFORM–PERCEPT⟩
    ⟨SENDER⟩ Simulator ⟨/SENDER⟩
    ⟨RECEIVER⟩ Agent01 ⟨/RECEIVER⟩
    ⟨ITERATION⟩ 2 ⟨/ITERATION⟩
    ⟨STATE⟩
        ⟨STREET⟩ 00 ⟨/STREET⟩
        ⟨CELL⟩ 00 ⟨/CELL⟩
        ⟨DIRECTION⟩ Right ⟨/DIRECTION⟩
        ⟨STATUS⟩ Still ⟨/STATUS⟩
    ⟨/STATE⟩
⟨/INFORM–PERCEPT⟩
```

And this process repeats as long as there is some agent in the simulator.

5.4 Description of the experiments

The NMDP framework and its algorithms for normative reasoning give us the chance to measure the impact of norms on the sequential decision-making of agents, allowing a sort of quantitative evaluations of the agents' performance. This section presents *comparative experiments*, where various experimental settings are applied to estimate the performance of *norm-compliant* and *self-interested* agents in distinct situations of the environment. The criteria by which the performance of individual agents (*micro* perspective) is measured are: (c1) the *computational resources* spent in the normative reasoning and policy construction; and (c2) the *average utility* gained in the motion simulator by executing its policy.

- According to criterion (c1), we say that the *norm-compliant* agent model overpasses the *self-interested* agent model if the norm-compliant consumes less computational resources than the self-interested during the reasoning processes.
- According to criterion (c2), we say that the *norm-compliant* agent model overpasses the *self-interested* agent model if the utility gained during the simulation is on average higher in norm-compliant agents than in self-interested agents.

Furthermore, from the *macro* perspective, the experiments assess the performance or effectiveness of various *penalties* and *norm enforcement intensities* (detection probabilities) in several populations of agents. This performance is estimated by means of a simulation (see Figure 5.4) using one of the following criteria: (c3) the percentage of agents that *crash* into each other along the simulation; or (c4) the average amount in *penalties* (sanctions) collected from the agents, that is, the amount of utility taken from the agents as a reaction against the norm violations.

Let $\lambda_i \mathsf{R_N}$ and $\lambda_j \mathsf{R_N}$ be penalties, and β_i and β_j be detection probabilities denoting norm enforcement intensities:

- According to criterion (c3), $(\lambda_i \mathsf{R_N}, \beta_i)$ outperforms $(\lambda_j \mathsf{R_N}, \beta_j)$ if the percentage of agents that crash into each other during the simulation is lower with $(\lambda_i \mathsf{R_N}, \beta_i)$ than with $(\lambda_j \mathsf{R_N}, \beta_j)$.
- According to criterion (c4), $(\lambda_i \mathsf{R_N}, \beta_i)$ outperforms $(\lambda_j \mathsf{R_N}, \beta_j)$ if the income from the imposition of sanctions (penalties collected from the agents) during the simulation is higher with $(\lambda_i \mathsf{R_N}, \beta_i)$ than with $(\lambda_j \mathsf{R_N}, \beta_j)$.

While a "good" set of norms according to the *crash* criterion (c3) clearly aims at helping the agents by minimizing the occurrence of crashes, we cannot affirm the same about the *penalty* criterion (c4). According to this last criterion (profit-oriented), a "good" set of norms maximizes the income from the imposition of penalties, which is dissociated from helping the agents to reach their destination gateways. For this reason, in some contexts, this type of profit-seeking motivation to design and enforce norms can be considered inappropriate. However, the appropriateness of the motivation behind the norms lies outside the scope of this thesis.

The simulations have been made under a variety of *experimental settings*, defined as follows on the basis of some initial runs:

(s1) λ: The penalty coefficient that multiplies $\mathsf{R_N}$ to determine the penalty imposed by the sanctions. The values of λ used in the experiments range from 0.01 to 0.30 with intervals of 0.01.

(s2) β: The detection probability of norm violations (norm enforcement intensity). As assumed in Section 5.1, the detection probability is the same, regardless of the state and the norm that has been violated in this state, that is, $\mathcal{D}(q, s) = \beta$ for any $q \in \mathcal{N}$ and $s \in \mathcal{S}$. The values of β used in the experiments range from 0.01 to 0.30 with intervals of 0.01.

(s3) μ: Population setting, which determines the properties of the active population of agents running the motion simulator, The population can be either *homogeneous*, composed of one agent model (self-interested or norm-compliant), or *heterogeneous*, composed of both agent models. The content of μ determines the number of agents per agent model. For example, $\mu = 10/20$ indicates that the population must be composed of 10 norm-compliant agents and 20 self-interested agents, totaling 30 agents.

(s4) ζ: Determines the algorithm to be employed in the policy construction process of the agents, which consists of searching for an exact solution (*non-stationary optimal policy*) to an infinite horizon MDP problem. Two algorithms have been employed to this end: Value Iteration [6] and Modified Policy Iteration [115, 95] (see Section 2.1.2 for details on these algorithms).

In the experiments throughout this chapter, $\mathsf{R_N} = 4.33$. Using this normalization constant, the average utility gained by the norm-compliant agent running in a single agent setting is equal to 1.0 (see Section 5.5.1). This way, we can easily visualize the difference between self-interested and norm-compliant agents in percentage terms, and establish a comparison between these models under different situations.

An experiment in the motion environment is carried out in three sequential phases, or processes, as illustrated in Figure 5.6:

(p1) *Construction of the agent classes*: In this phase, we construct the *agent classes*, which will be employed in third phase to create the *agents* to be launched in the motion simulator. At this point, it is essential to distinguish *agent model* from *agent class*. The notion of agent model concerns to the internal reasoning processes that determine the behaviour of an agent. In this thesis, there are two agent models, norm-compliant and self-interested. An agent class, on the other hand, is a construct used to create agents, which enables them to have not only behaviour, but state as well. That is, an agent class consists of an agent models' implementation that determines its behaviour, and an NMDP

instance, described in Section 5.2, that contains the agent's knowledge and determines its destination gateway where it receives the positive reward. In the experiments presented in this chapter, there are 16 agent classes resulting from the combination of 2 agent models and 8 possible destination gateways. Notice that the penalty coefficient λ (s1) and the detection probability β (s2) complete the specification of the norms and detection function, as the agents are supposed to know these settings. By creating new agents and removing the ones that have ended or crashed, we maintain an *active population* running in the motion simulator (this is detailed in the third phase).

(p2) *Reasoning*: In this phase, each agent class constructed in (p1) autonomously computes an *optimal policy* by reasoning over its NMDP instance. The normative reasoning processes are determined by the agent model, while the policy construction algorithm is indicated in the MDP solver setting ζ (s4). The reasoning processes are performed offline, that is, each agent computes its individual policies before it joins the motion simulator. In this second phase, we measure the *computational resources* (the agents' state space size and computation time) consumed in the reasoning processes. Notice that the outcome of this phase is a set of optimal policies (one per agent class) regardless the algorithm determined in ζ (s4), Thus, ζ (s4) impacts only on the consumption of computational resources. Section 5.5.4 will present and discuss the results of this phase.

(p3) *Simulation*: This phase consists of launching agents in the motion simulator so as to measure the *utility*, *crashes* and *penalties* of individual agents or certain groups of agents. The agents running in the motion simulator compose the *active population* generated on the basis of the *population setting* μ (s3). The penalty coefficient λ (s1) and the detection probability β (s2) complete the specification of the detection function and the norms that will regulate the simulation. The results of this phase will be discussed from Section 5.5.1 to Section 5.5.3.

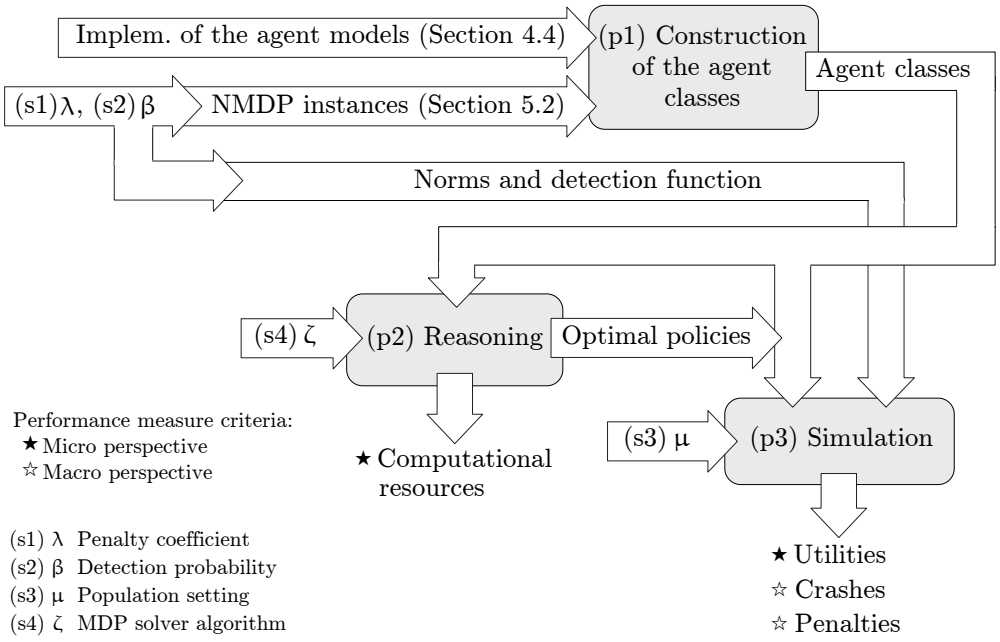


Figure 5.6: Phases of an experiment, illustrated as grey boxes, and their respective input and output data specified within the arrows. The black star denotes the criteria by which the performance of individual agents (micro perspective) is measured, while the white star denotes the criteria by which the effectiveness of the norms is measured.

Figure 5.7 details the sub-processes that compose the simulation phase (p3): the *motion simulator*, already introduced in Section 5.3, the *active population* of agents running in the simulator, the *population manager*, which launches and removes agents to keep the active population operating in the simulator, and finally, the *graphical interface* that illustrates the simulation as it advances on the time. Whenever an agent leaves or crashes in the simulated motion environment, it receives an inform-termination message from the simulator (see Section 5.3 for details on the communication protocols between the agents and the simulator). At this moment, the population manager replaces this agent by a new one of the same model in order to keep the population size per agent model as determined in the *population setting* μ (s3). The NMDP instance, which determines the destination gateway, as well as the origin gateway (the agent's initial state), are randomly selected by the population manager: the probability of choosing any destination gateway is 1/8 (remind that there are 8 gateways) and the probability of choosing any origin gateway, excluding the selected destination gateway, is 1/7. This way, we obtain an uniform distribution whereby any

combination (without repetition) of origin and destination is equally likely to happen. However, the uniformity of the traffic flow cannot be ensured given that the agents are autonomous to decide their own routes and destinations. Once the population manager has chosen the agent model and the destination gateway, it constructs the new agent from the corresponding agent class, and assigns to this agent the chosen origin gateway as the initial state and its corresponding policy already computed during the second phase (p2). Finally, the newborn agent autonomously sends a request-for-join message to the simulator requesting to join the simulated environment. If the agent is accepted by the simulator, it operates in the environment until it crashes or leaves. At this moment, the agent is replaced by a new one.

The simulation ends when the population manager stops to create new agents and the agents in the active population end. The interruption of the population manager can be made directly by the user, or alternatively, the number of agent replacements can be determined prior to the simulation. In this case, when the population manager reaches the number of agent replacements, it no longer creates new agents.

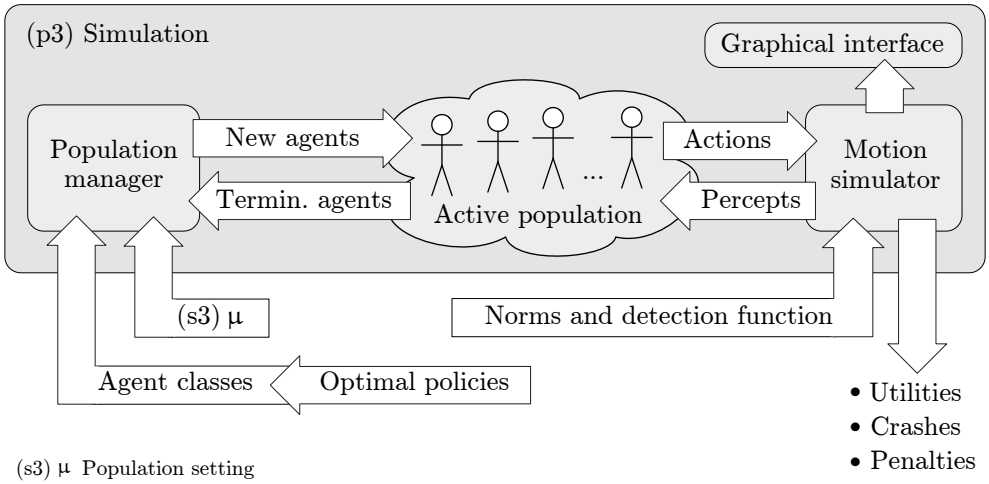


Figure 5.7: Simulation phase (p3) composed of the following sub-processes: *population manager*, *motion simulator*, *graphical interface* and *active population* of agents. The arrows denote data flow between the processes.

The experiments are organized as follows:

(e1) *Single agent experiments*:

The active population must have one agent:

(s3) $\mu = 1/0$ (one norm-compliant agent) or

(s3) $\mu = 0/1$ (one self-interested agent).

As there is only one agent at time in the motion simulator, we can affirm that the agents' knowledge about the environment is complete and precise, thus, the expected utility calculated by the agent expresses with accuracy the average utility gained in the simulator. Furthermore, since these agents run alone in the motion simulator (one enters only another exits), crashes never happen. These experiments measure the performance of the agent models (micro perspective) by means of the utilities they gain in the motion simulator (c2). The experiments have been executed under a range of penalties and norm enforcement intensities,

(s1) $\lambda \in [0.01, 0.30]$ and

(s2) $\beta \in [0.01, 0.30]$.

This way, we can measure and compare the performance of the agents in several situations. From the macro perspective, the sum of penalties collected (c4) is measured. Section 5.5.1 will present and discuss the results.

(e2) *Multiagent experiments with homogeneous populations*:

The active population is composed of multiple agents of the same agent model:

(s3) $\mu = x/0$ (multiple norm-compliant agents), where $x > 1$, or

(s3) $\mu = 0/x'$ (multiple self-interested agents), where $x' > 1$.

In settings with norm-compliant agents, crashes never occur as the compliance with the norms ensures the absence of coordination problems. On the other hand, in settings with self-interested agents, crashes may happen as long as the agents violate some norm. Here, the agents' knowledge about the motion environment is incomplete: as previously mentioned, the agents know nothing about their acquaintances. These experiments measure the performance of the agents by means of the utilities they gain in the motion simulator (c2). A range of penalties and norm enforcement intensities,

(s1) $\lambda \in [0.01, 0.30]$ and

(s2) $\beta \in [0.01, 0.30]$,

has been used in the experiments, so we can estimate and compare the agents' performance under different situations. With self-interested agents, the percentage of agents that crash (c3) and the sum of penalties imposed by sanctions (c4) are measured as well. Section 5.5.2 will present and discuss the results.

(e3) *Multiagent experiments with heterogeneous populations*:
The active population is composed of multiple agents of both agent models:
(s3) $\mu = x/x'$, where $x \geq 1$ and $x' \geq 1$.
As in (e2), these experiments measure the performance of the agent models by means of the utilities they gain in the motion simulator. The experiments have been carried out under a range of penalties and norm enforcement intensities,
(s1) $\lambda \in [0.01, 0.30]$ and
(s2) $\beta \in [0.01, 0.30]$,
so we can estimate the performance of the agents (c2) in several situations. Here, self-interested agents and norm-compliant agents are launched together in the simulator, and both agent models are subject to crashes as long as some self-interested agent violates some norm. From the macro perspective, we compare the percentage of agents that crash by agent model (c3), and we sum the penalties imposed by the sanctions addressed to the self-interested agents (c4). With these multiagent simulations with heterogeneous populations, it is expected to understand to what extent the behaviour of one agent model affects the performance of the other in different environment situations. The results of these experiments are presented and discussed in Section 5.5.3.

Finally, the assessment of computational resources spent in the reasoning processes (p2), common to the experiments (e1), (e2) and (e3), is presented in Section 5.5.4.

5.5 Results

5.5.1 Single agent experiments

These are the simplest experiments since there is only one agent at time in the simulator:

(s3) $\mu = 1/0$ (one norm-compliant agent) or
(s3) $\mu = 0/1$ (one self-interested agent).

The experiments have been executed under a range of penalties and norm enforcement intensities,

(s1) $\lambda \in [0.01, 0.30]$ and
(s2) $\beta \in [0.01, 0.30]$.

Figure 5.8 shows that the average utility gained by a norm-compliant agent running alone in the motion environment is *constant*. This is explained by the fact that this agent model always follows the norms, no matter the penalty coefficient λ (s1) or detection probability β (s2). The non-normalized average utility is $0.2309\mathsf{R_N}$, and the value 1.0 has been obtained by setting $\mathsf{R_N} = 4.33$. This normalization constant is used throughout this chapter, so we can easily visualize the difference of performance between any agent and this norm-compliant agent in percentage terms.

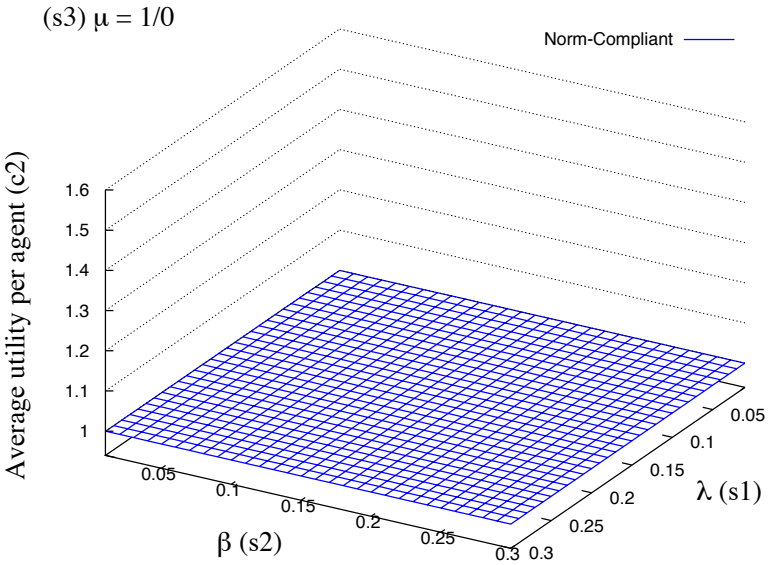


Figure 5.8: Average utility gained by the norm-compliant agent running alone in the motion environment – (s3) $\mu = 1/0$.

In the absence of norms regulating the road segments, the optimal choice would be taking the shortest path to the destination gateway. However, in such an environment ruled by norms, a self-interested agent takes the sanctions into account to maximize its utility. Figure 5.9 shows the average utilities gained by a self-interested

agent running alone in the simulator. As the penalty coefficient λ (s1) and the detection probability β (s2) increase, the average utility decreases until it reaches 1.0 (the same average utility gained by the norm-compliant model). At this moment, the self-interested agent decides to obey the norms because violating them no longer exceeds the expected utility of complying with them. So, in the single agent motion environment, where crashes are not possible, the norm-compliant agent never outperforms the self-interested agent with respect to the average utility gains (c2).

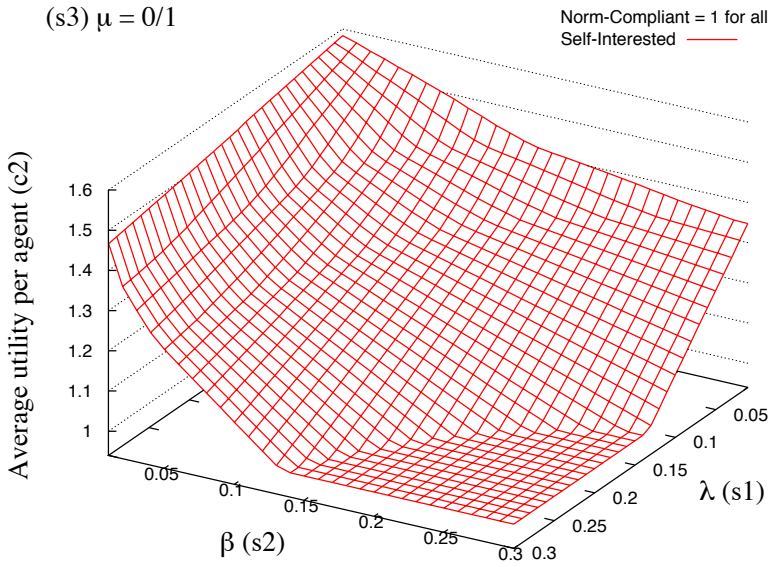


Figure 5.9: Average utility gained by the self-interested agent running alone in the motion environment – (s3) μ = 0/1. Each point in the surface gives the average utility under a particular penalty coefficient λ (s1) and detection probability β (s2).

In the experiments on which the self-interested agent has outperformed the norm-compliant agent according to the *average utility* criterion (c2), the self-interested agent has violated some norm (i.e the norm regulating the traffic direction in the street) in order to take a shorter path to the destination gateway. In these situations, the cost saved by taking the non norm-compliant shorter path is larger than the cost imposed by the sanctions. Figure 5.10 shows the average percentage of the lifetime of the self-interested agents spent in norm violating states under a particular penalties λ (s1) and detection probabilities β (s2). As λ and β decrease, the time spent in norm violating

states increase. Starting from the bottom of the chart (λ=0.30, β=0.30), there is an area which corresponds to combinations of λ and β that ensure the absence of norm violations (0.0 per cent). Then, the chart shows an increment on the norm violations, up 7.6 per cent, followed by a stabilization on the values. Finally, the norm violations rise again towards ($\lambda = 0.01, \beta = 0.01$), up to 41 per cent.

Notice that the percentage of the agents' lifetime spent in norm violating states is not adopted as a performance criterion. Here, this data is used to explain where the utility gains of the self-interested agents, shown in Figure 5.9, come from.

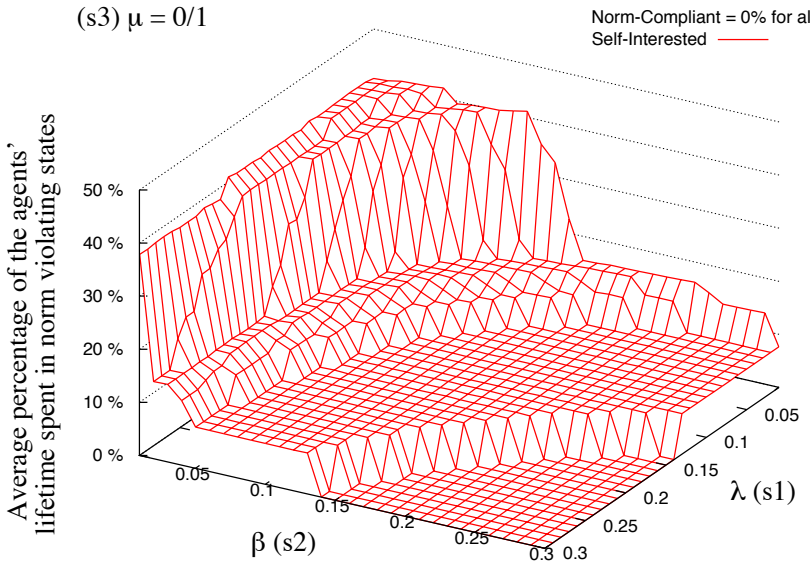


Figure 5.10: Average percentage of the lifetime of the self-interested agents spent in norm violating states. Each point in the surface gives the average percentage under a particular penalty coefficient λ (s1) and detection probability β (s2).

From the *macro perspective*, the experiments (e1) estimate the effectiveness of various penalties and norm enforcement intensities by means of the average income from *penalties* collected in a given period of time (c4). Figure 5.11 illustrates the average amount in penalties collected in a period of 1000 time-steps as a reaction against the norm violations. Each point in the chart corresponds to the utility taken under a particular penalty coefficient λ (s1) and detection probability β (s2). Notice that the chart has been rotated, so the entire surface can be visualized.

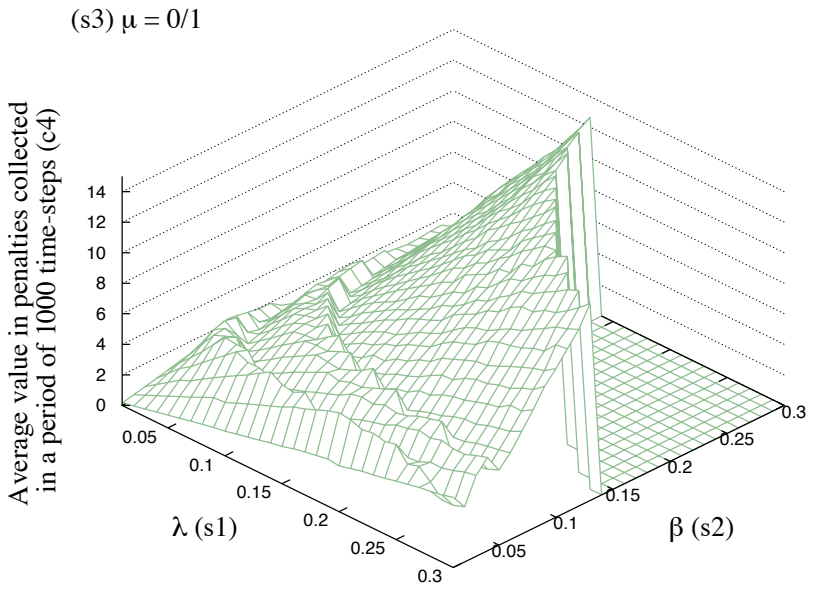


Figure 5.11: Average value in penalties collected in a period of 1000 time-steps. Each point in the surface gives the average value under a particular λ (s1) and β (s2).

Analyzing Figure 5.11, we can observe that high penalties and norm enforcement intensities result in no income as these settings inhibit any violation of norms[4]. As λ and β decrease (from the right side to the left side of the chart) we can see an abrupt rise on the values. After this, the values tend to decrease as λ and β decrease. The maximum value (13.17) has been obtained with $\lambda = 0.16$ and $\beta = 0.25$.

Figure 5.12 shows in the same chart the average percentage of the self-interested agents' lifetime spent in norm violating states (in red, left y-axis) and the average amount in penalties collected in a period of 1000 time-steps (in green, right y-axis). The x-axis indicates the norm enforcement intensity β (s2) for the fixed coefficient $\lambda = 0.20$ (s1) arbitrarily chosen. This chart is analyzed in different intervals of β:

- $[0.01, 0.02]$ and $[0.04, 0.06]$: As β increases, the norm violations decrease and the income from penalties increases. In this case, the gain caused by the ascent on β compensates the loss caused by the descent on the number of norm violations.

[4]Notice that the area in Figure 5.11 where there is no income from penalties corresponds to the area in Figure 5.10 where there is no norm violations, and corresponds to the area in Figure 5.9 where the agent's utility is 1.0 (the self-interested agent assumes a norm-compliant behaviour).

- $[0.02, 0.03]$, $[0.06, 0.07]$ and $[0.20, 0.21]$: As β increases, the norm violations decrease and the amount in penalties collected decreases as well. Here, the gain caused by the ascent on β do not compensate the loss caused by the descent on the number of norm violations.
- $[0.03, 0.04]$ and $[0.07, 0.20]$: The quantity of norm violations is stable in these intervals, and the income from penalties rises as β increases.
- $[0.21, 0.30]$: The norm enforcement intensity is high enough to inhibit any norm violating behaviour. In this case, there is no penalties to be collected.

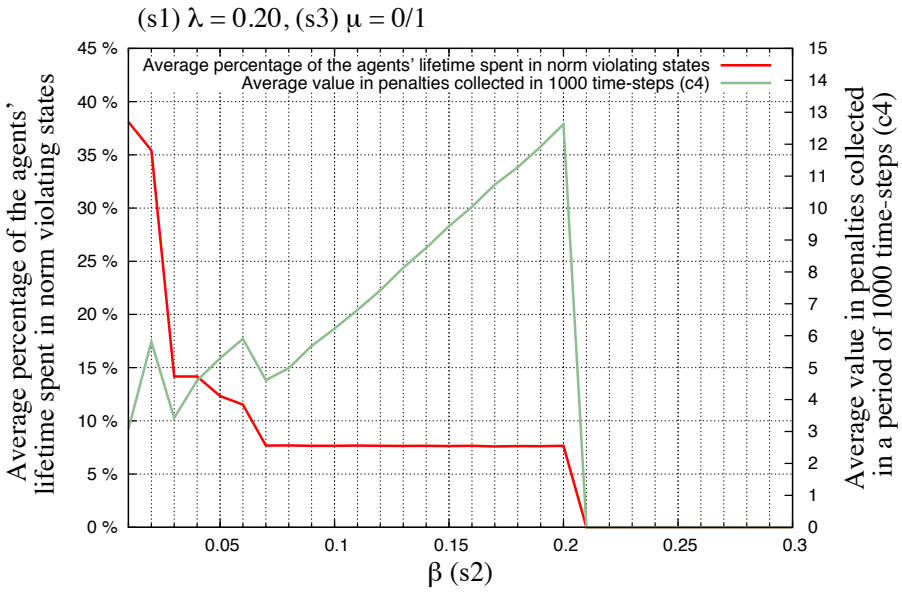


Figure 5.12: Average percentage of the agents' lifetime spent in norm violating states (left y-axis) and the average value in penalties collected in 1000 time-steps (right y-axis). The x-axis indicates the norm enforcement intensity β (s2). The experimental setting is completed with $\lambda = 0.20$ (s1) and $\mu = 0/1$ (s3).

As we can observe, the percentage of the agents' lifetime spent in norm violation is a monotonic function – it does not increase as β (or λ) increases. However, we cannot affirm the same about the income from penalties collected from the agents, which is determined not only by β and λ, but also by the quantity of norm violations, which in turn, is determined by the autonomous agents on the basis of β and λ. Example 8 shows how these parameters determine the income from penalties.

Example 8. *Consider the following three combinations of* β *and* λ*:*

$$(\lambda = 0.20, \beta = 0.05),$$
$$(\lambda = 0.20, \beta = 0.06),$$
$$(\lambda = 0.20, \beta = 0.07).$$

Under these combinations, the self-interested agents stay on average 12.2%*,* 11.5% *and* 7.6% *of its lifetime in norm violating states, respectively. In* 1000 *time-steps, it corresponds to* 122*,* 115 *and* 76 *time-steps occupying norm violating states. Using the respective* β*, we can calculate how many times, on average, the norm violations are detected and the norm violators are punished:*

$$122 \times 0.05 = 6.10,$$
$$115 \times 0.06 = 6.90,$$
$$76 \times 0.07 = 5.32.$$

Then, we multiply these values by $\lambda \mathsf{R}_\mathsf{N}$ *in order to calculate the income from sanctions in a period of* 1000 *time-steps* ($\lambda = 0.20, \mathsf{R}_\mathsf{N} = 4.33$)*:*

$$6.10 \times 0.20 \times 4.33 = 5.28,$$
$$6.90 \times 0.20 \times 4.33 = 5.97,$$
$$5.32 \times 0.20 \times 4.33 = 4.61.$$

In conclusion, if we increase the enforcement intensity β *from* 0.05 *to* 0.06*, the income increases from* 5.28 *to* 5.97*. However, if we increase to* 0.07*, the income decreases to* 4.61 (*see Figure* 5.12)*. In both cases, the norm violations decrease.*

Example 8 shows that the income collected from penalties is a function of λ, β and the quantity of norm violations. However, only λ and β are determined by the entity that imposes the norms. The quantity of norm violations is determined by the autonomous agents. Increasing λ and β means *higher income per sanction* and *higher percentage of norm violations detected*, respectively. But, increasing λ and β also means that the agents tend to deviate less from the norms. In conclusion, according to the criterion (c4), to increase λ and β is a good deal only if such an ascent on these values compensates the loss caused by the change in the agents' behavior.

5.5.2 Multiagent experiments with homogeneous populations

In these experiments, the active population is composed of multiple agents of the same agent model (homogeneous population):

(s3) $\mu = x/0$ (multiple norm-compliant agents), where $x > 1$, or
(s3) $\mu = 0/x'$ (multiple self-interested agents), where $x' > 1$.

The experiments have been executed under a range of penalties and norm enforcement intensities,

(s1) $\lambda \in [0.01, 0.30]$ and (s2) $\beta \in [0.01, 0.30]$.

In the motion environment, the norms are reasonable designed in the sense that the addressee agents are capable of achieving any destination gateway by fully complying with the norms, and this fully compliance by the whole population guarantees the absence of crashes and impositions of sanctions. In this context, the average utility gained by an agent in norm-compliant homogeneous populations is constant (equal to 1.0) for any penalty coefficient λ (s1), norm enforcement intensity β (s2) or population size x (s3). Thus, the utility chart of individual agents in norm-compliant homogeneous populations is identical to the chart in Figure 5.8.

On the other hand, in self-interested populations, as long as some agent violates some norm, their individual utility is affected not only by the imposition of sanctions to punish the norm violations, but also by crashes. Figure 5.13 specifies the average utility per agent in a homogeneous population of 10 self-interested agents, $\mu = 0/10$ (s3), under different combinations of λ (s1) and β (s2). Starting from the right side of the chart, where λ=0.30/β=0.30 (notice that the chart has been rotated, so the entire surface can be visualized), there is an area where the utility is equal to 1.0. In this area, the penalty coefficient λ (s1) and the detection probability β (s2) are high enough to force the self-interested agents to follow the norms. Then, the utility drops abruptly as a result of the norm violations (imposition of penalties) and crashes. Notice that this abrupt fall coincides with the first rise of norm violations and crashes (see Figure 5.10 and Figure 5.14). After such an abrupt fall, as long as the amount of norm violations and crashes stay constant, the utility rises as the λ and β decrease. Finally, the average utility drops again as a consequence of successive increments on the norm violations and crashes. Using the utility criterion (c2), in the experiments (e2), the *norm-compliant* agents outperform or perform as well as the *self-interested* agents under any combination of λ (s1) and β (s2).

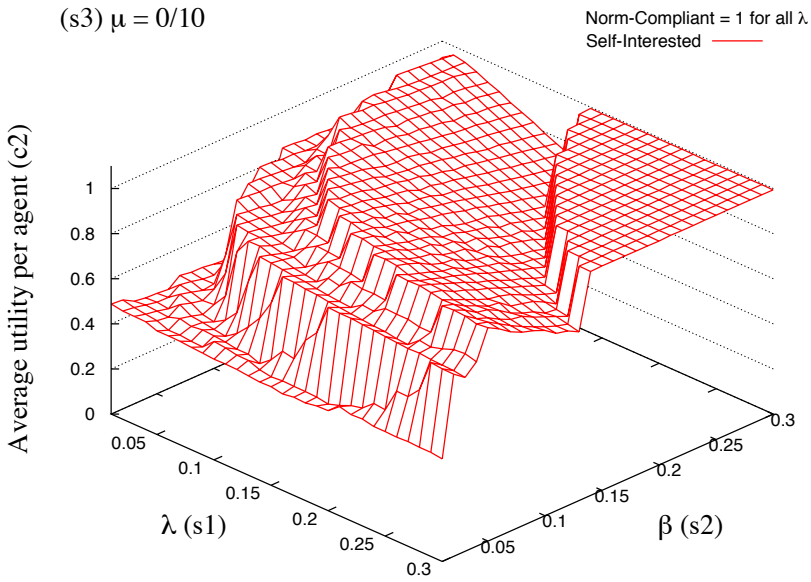


Figure 5.13: Average utility per agent in a population of 10 self-interested agents – (s3) $\mu = 0/10$. Each point in the surface gives the average utility under a particular penalty coefficient λ (s1) and detection probability β (s2).

As previously said, the agents have no information about their acquaintances. In this context, crashes between the agents may happen as long as some of them violate the norms. From the *macro perspective*, the experiments estimate the percentage of agents that crash (c3) and the income from the collection of penalties against norm violations in self-interested populations (c4).

Figure 5.14 shows the percentage of agents that crash as a consequence of norm violations. Starting from the bottom of the chart, where λ=0.30/β=0.30, we observe a flat area with no crashes. Under these setups, the agent obeys the full set of norms. Then, an increment is observed, followed by a stabilization of the values. Finally, we can see successive increments on the percentage.

Observe that the surface form of this chart matches the surface form of the chart in Figure 5.10, indicating that the percentage of agents that crash is directly proportional to the percentage of the agents' lifetime spent in norm violating states. Thus, according to the criterion (c3), the largest values of λ (s1) and β (s2) are the best – enforcing fully norm-compliance ensures the absence of crashes.

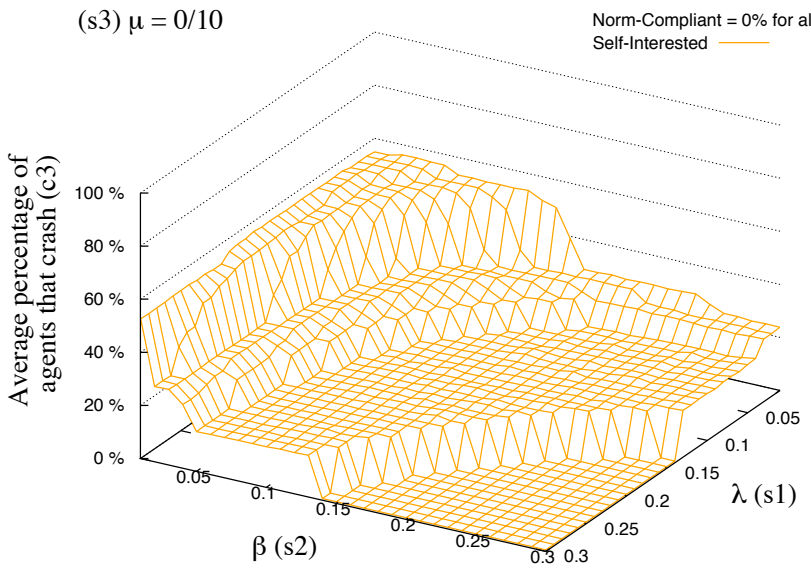


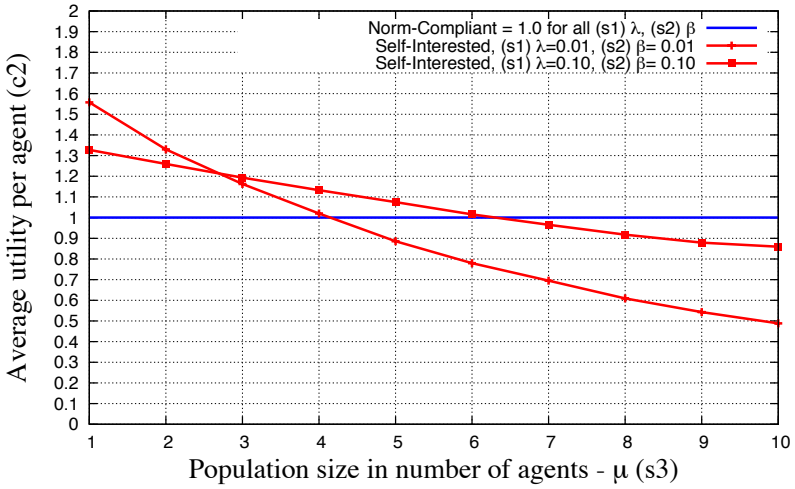
Figure 5.14: Average percentage of agents that crash in a population of 10 self-interested agents – (s3) $\mu = 0/10$. Each point in the surface gives the percentage under a particular penalty coefficient λ (s1) and detection probability β (s2).

A correlation between *crashes* and *average utility* can be observed in Figure 5.15, which presents: (a) the average utility of the agents and (b) the average percentage of agents that crash. These experiments have been carried out with several population sizes (s3), ranging from $\mu = 0/1$ to $\mu = 0/10$, and with two fixed combinations of λ (s1) and β (s2):

- (s1) $\lambda = 0.01$ and (s2) $\beta = 0.01$; and
- (s1) $\lambda = 0.10$ and (s2) $\beta = 0.10$.

Figure 5.15 (a) shows that the average utility of the self-interested agents (red lines) declines as the population of self-interested agents increases, while the average utility of the norm-compliant agents (blue line) is constant. The descent on the self-interested agents' utilities is caused by the ascent on the percentage of agents that crash, illustrated in Figure 5.15 (b). According to the *utility criterion* (c2), if (s1) λ=0.01 and (s2) β=0.01, the self-interested agents outperform the norm-compliant agents only in populations smaller than 5 agents; if (s1) λ=0.10 and (s2) β=0.10, the self-interested agents outperform the norm-compliant agents only in populations smaller than 7 agents.

(a)



(b)

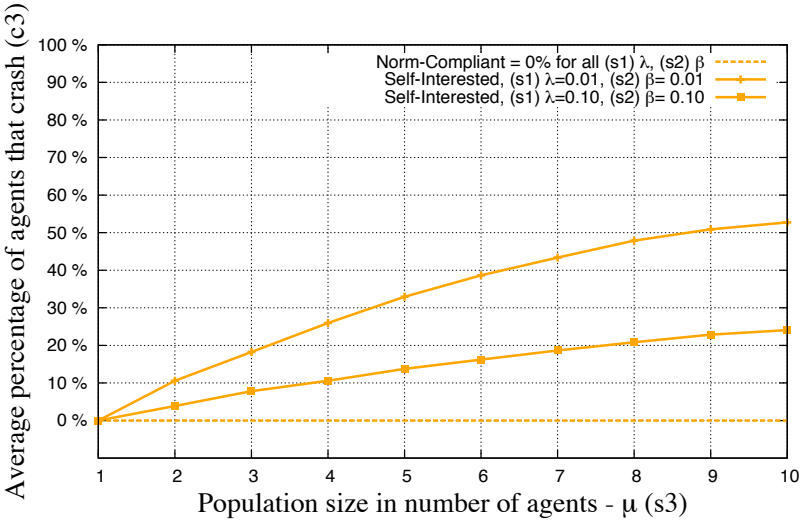


Figure 5.15: Graphs showing (a) the average utility of the individual agents and (b) the average percentage of agents that crash with several population sizes ranging from 1 to 10 agents (s3).

Still, from the *macro perspective*, the experiments (e2) estimate the effectiveness of several penalties and norm enforcement intensities with respect to the income from penalties collected in a given period of time (c4). Figure 5.16 shows the average amount in penalties collected from 10 self-interested agents, (s3) $\mu = 0/10$, during 1000 time-steps. Each point in the chart indicates the utility taken under a particular λ (s1) and β (s2). The surface of the chart in Figure 5.16 has the same form (shape) of the surface of the chart in Figure 5.11. The only difference between these charts is the income (z-axis), which is $\sim$10 times larger in the last for any λ and β. This indicates that the income from penalties rose proportionally to the number of agents running in the motion environment. The maximum value (141.36) has been obtained with $\lambda = 0.16$ and $\beta = 0.25$ as well.

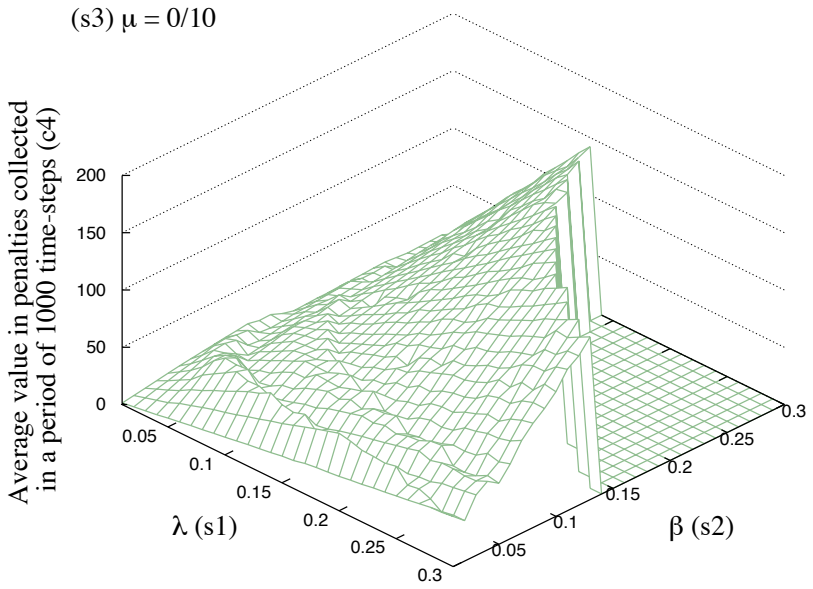


Figure 5.16: Average value in penalties collected in a period of 1000 time-steps from a population of 10 self-interested agents – (s3) $\mu = 0/10$. Each point in the surface gives the average income under a particular λ (s1) and β (s2).

5.5.3 Multiagent experiments with heterogeneous populations

In the multiagent experiments with heterogeneous populations, the active population is composed of multiple agents of both agent models:

(s3) $\mu = x/x'$, where $x \geq 1$ and $x' \geq 1$.

Three combinations of penalty coefficients (s1) and norm enforcement intensities (s2) have been used to create several situations of the environment:

- (s1) $\lambda = 0.01$ and (s2) $\beta = 0.01$,
- (s1) $\lambda = 0.10$ and (s2) $\beta = 0.10$, and
- (s1) $\lambda = 0.20$ and (s2) $\beta = 0.20$.

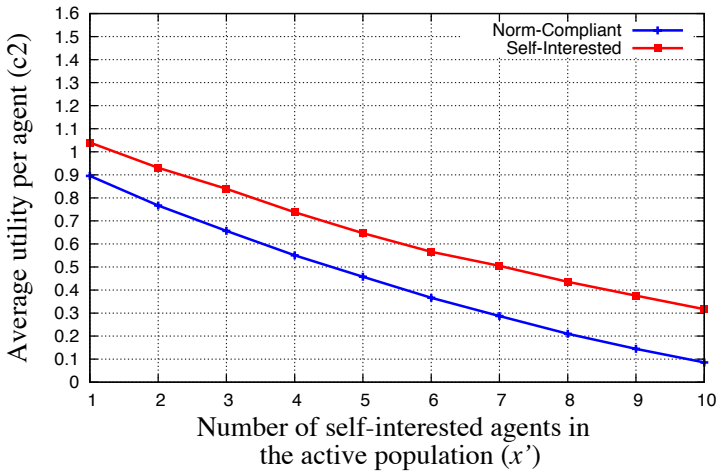
The previous experiments have shown that these three combinations produce different levels on the amount of norm violations by self-interested agents, which in turn, impacts on the agents' utility (c2), amount of crashes (c3) and income collected from penalties imposed against norm violations (c4). This makes these settings adequate to produce different situations of the environment.

Here, self-interested agents and norm-compliant agents are launched together in the motion simulator, and both agent models are subject to crashes as long as some self-interested agent violates some norm. This means that the behaviour of the self-interested agents may worsen the performance of the norm-compliant agents.

From the micro perspective, Figure 5.17 shows the average utility per agent with the population setting fixed to $\mu = 5/x'$, that is, the number of norm-compliant agents in the active population is fixed to 5, so we can check the impact of varying the number of self-interested agents. In Figure 5.17 (a) we set a low penalty and norm enforcement intensity, (s1) $\lambda = 0.01$ an (s2) $\beta = 0.01$, while in Figure 5.17 (b) we set a high penalty and norm enforcement intensity, (s1) $\lambda = 0.20$ an (s2) $\beta = 0.20$. In both cases, the utilities of all agents fall as the number of self-interested agents increase. The difference is that with the lower penalty the utilities fall in higher rates. We can also observe that with the lower penalty, the self-interested agents outperform the norm-compliant agents, and with higher penalty, the norm-compliant agents outperform the self-interested agents.

(s1) λ = 0.01
(s2) β = 0.01
(s3) μ = 5/x'

(a)



(s1) λ = 0.20
(s2) β = 0.20
(s3) μ = 5/x'

(b)

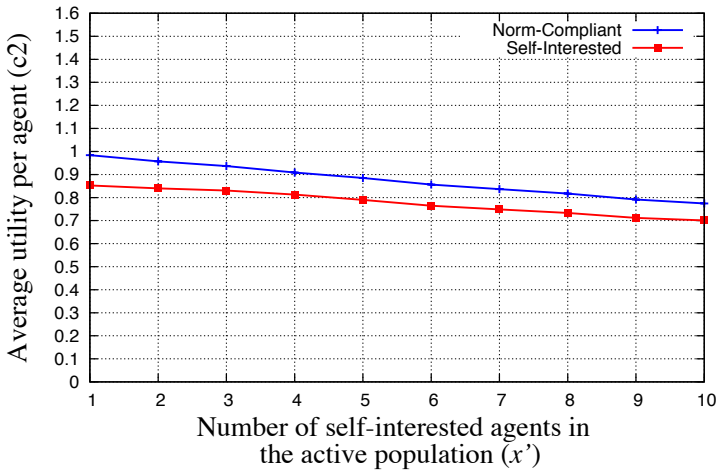
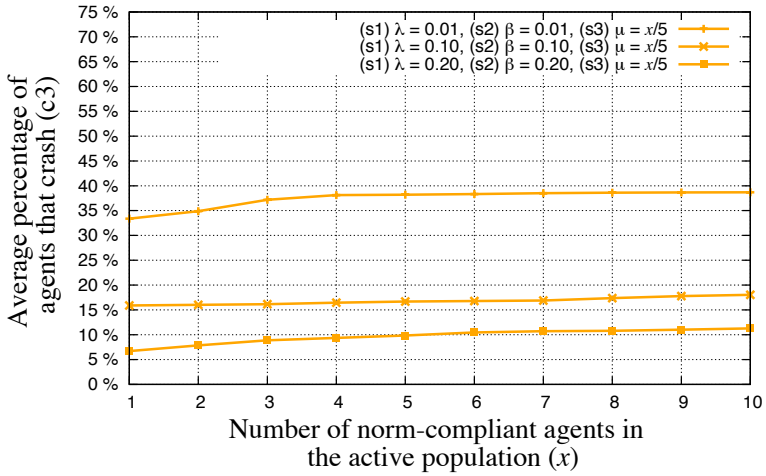


Figure 5.17: Average utilities of agents in a heterogeneous society.

From the macro perspective, Figure 5.18 shows the average percentage of agents that crash with different settings. In Figure 5.18 (a) the population setting is $\mu = x/5$, that is, the number of self-interested agents in the active population is fixed to 5, and the number of norm-compliant agents in the active population varies as indicated in the x-axis. As we can see, as the crashes rise slightly as the number of norm-compliant agents grows. On the other hand, in Figure 5.18 (b), we can observe steeper rises on the crashes. In this chart, the population setting is $\mu = 5/x'$, that is, the number of norm-compliant agents in the active population is fixed to 5, and the number of self-interested agents in the active population varies as indicated in the x-axis. The

steeper rises are explained by the fact that the higher the number of self-interested agents violating norms, the more likely that any agent in the environment crashes. In Figure 5.18 we can also observe that the higher λ (s1) and β (s2), the smaller the average number of agents that crash.

(a)



(b)

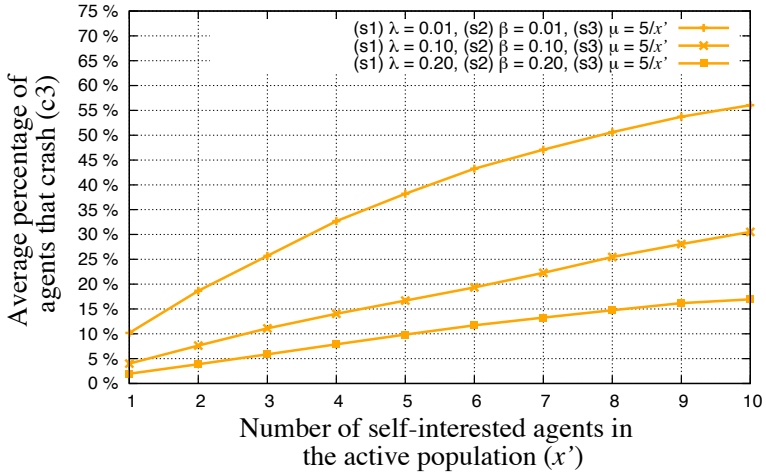


Figure 5.18: Average percentage of agents that crash in a heterogeneous population with different settings.

Still, from the macro perspective, the experiments (e3) estimate the average income from penalties against norm violations (c4), collected in a given period of time from heterogeneous populations. Figure 5.19 shows the average value in penalties collected in 1000 time-steps from heterogeneous populations composed of 5 norm-compliant agents and x' self-interested agents, $\mu = 5/x'$ (s3). As we can observe, for any of the three combinations of λ (s1) and β (s2), the income rises as the number of self-interested agents in the active population grows. That is, the larger the number of agents violating the norms, the larger the income. On the other hand, varying the number of norm-compliant agents in the active populations has shown no increase on the income as these agents are never penalized.

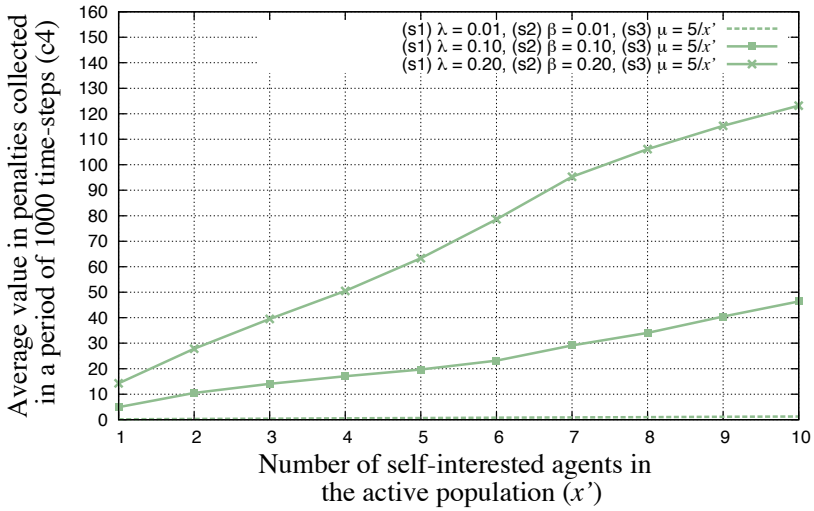


Figure 5.19: Average value in penalties collected in a period of 1000 time-steps from a heterogeneous population of 5 norm-compliant agents and x' self-interested agents.

5.5.4 Resource consumption

This section empirically estimates the resource consumption of the *reasoning processes* of the agent models. To this end, the time consumed by the algorithms is measured with the Java Interactive Profiler (JIP)[5], a high performance, low overhead profiler that is written entirely in Java and distributed under the BSD license (freeware). The output of this tool is provided and explained in Appendix C.

As we can see in Figure 5.6, the reasoning phase (p2) has two inputs: the *agent classes* (see the construction of the agent classes (p1) in Section 5.4) and ζ (s4), an algorithm to construct policies for MDPs.

Two algorithms have been experimented:

(i) ζ = Value Iteration (VI), and

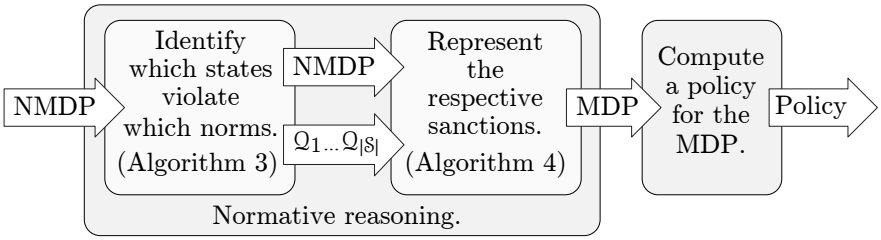
(ii) ζ = Modified Policy Iteration (MPI).

Figure 5.20 illustrates the internal processes of the agent models, which provide the structure by which the remaining of this section is organized:

- Firstly, this section estimates the time spent by the *normative reasoning* of the agent models so as to establish a comparison between them. More specifically, it compares Algorithm 3 with Algorithm 5, and Algorithm 4 with Algorithm 6.
- Then, it calculates the size of the state space of the MDP that results from the normative reasoning. The size of the state space is related to the time necessary to compute an optimal policy: the smaller the state space, the faster the policy construction tends to be.
- Finally, this section presents the results on the policy construction.

[5] http://jiprof.sourceforge.net

(a) Self-interested agent model



(b) Norm-compliant agent model

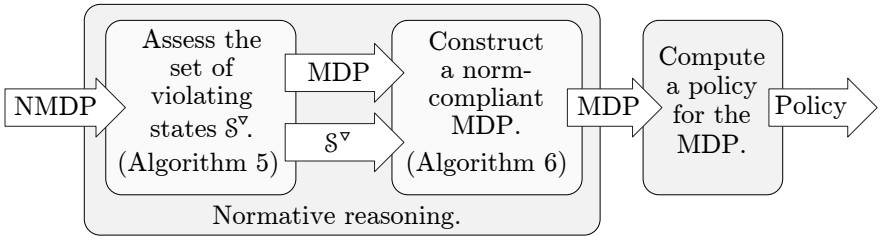


Figure 5.20: (a) Self-interested agent model and (b) norm-compliant agent model. For details on the agent models, see Section 4.2 and Section 4.3, respectively.

Table 5.2 specifies the time (in milliseconds) consumed in the *normative reasoning* of the agent models. As we can see, Algorithm 5 has been 1.04 times faster than Algorithm 3, and Algorithm 6 has been 3.48 times faster than Algorithm 4. Considering the total time spent in the normative reasoning, the norm-compliant agent model has been ∼1.35 times faster than the self-interested agent model.

Table 5.2: Time in milliseconds consumed in the normative reasoning.

Agent model	Algorithm	Time (ms)	Total time (ms)
Self-interested	Algorithm 3	2856,3	4297,7
	Algorithm 4	1441,4	
Norm-compliant	Algorithm 5	2746,4	3160,4
	Algorithm 6	414,0	

The time spent by Algorithm 3 and Algorithm 5 has been determined to a greater extent by the Java Expression Parser (JEP)[6], a Java library for parsing and evaluating mathematical expressions, which is used in the normative reasoning to determine if a state satisfy the normative context/content of a norm (boolean satisfiability problem). In Algorithm 3, the parsing and evaluation of boolean expressions correspond to ∼97% of the time, while in Algorithm 5, these procedures represent ∼88% of the time. See Appendix C for further details on the assessment of this data.

The NMDP instances specified in Section 5.2 contain 3200 states, resulting from the combination of the features: 16 values for STREET, 25 for CELL, 4 for DIRECTION, and 2 for STATUS. Out of these 3200 states, 50% are invalid combinations of STREET and CELL, which are unreachable from any valid state by means of state-transitions. The valid combinations,

$$\{00, 01, \ldots 07\} \times$$
$$\{00, 01, \ldots 23\} \times$$
$$\{\text{UP, RIGHT, DOWN, LEFT}\} \times$$
$$\{\text{STILL, MOVING}\} = 1536 \text{ states, and}$$

$$\{\text{NW, NE, SW, SE, WN, WS, EN, ES}\} \times$$
$$\{\text{GATEWAY}\} \times$$
$$\{\text{UP, RIGHT, DOWN, LEFT}\} \times$$
$$\{\text{STILL, MOVING}\} = 64 \text{ states,}$$

sum 1600 states, which correspond to the agents' working state space. While the self-interested agents explore the entire working state space to construct their policies, the norm-compliant agents explore only part of it – the subset of states that comply with all norms. By excluding the states that violate some norm, the working state space of the norm-compliant agents is reduced from 1600 to 432 states, a reduction of 73%. The number of admissible actions per state of the norm-compliant agents has been reduced as well. While the self-interested agents have to decide among 4 actions on average, the norm-compliant agents have to choose from 2 actions on average. Of course, the magnitude of these reductions depends on the specification (design) of the norms, since the norm-compliant states are determined on this basis.

[6]The experiments performed in this thesis use the version 2.4.1 of JEP, distributed under the GPL license. The developer states that the latest version (3.4) parses and evaluates expressions faster than the past releases. However, the versions higher than 2.4.1 are distributed under a commercial license (payware), and for this reason, they have not been tested in this thesis. More information about the library can be found in developer's page: http://www.singularsys.com/jep/

To study the relation between the size of the state space of each agent model and the time consumed by them to construct an *optimal policy*, two algorithms for solving MDPs (s4) have been tested: Value Iteration (VI) [6] and Modified Policy Iteration (MPI) [95]. In the VI version used here, described in Chapter 2, the error tolerance is $\epsilon = 10^{-3}\mathsf{R_N}$ and the discount factor is $\gamma = 0.99$. With the MPI, described in Chapter 2 as well, a series of initial runs have been made to identify an effective configuration, detailed later.

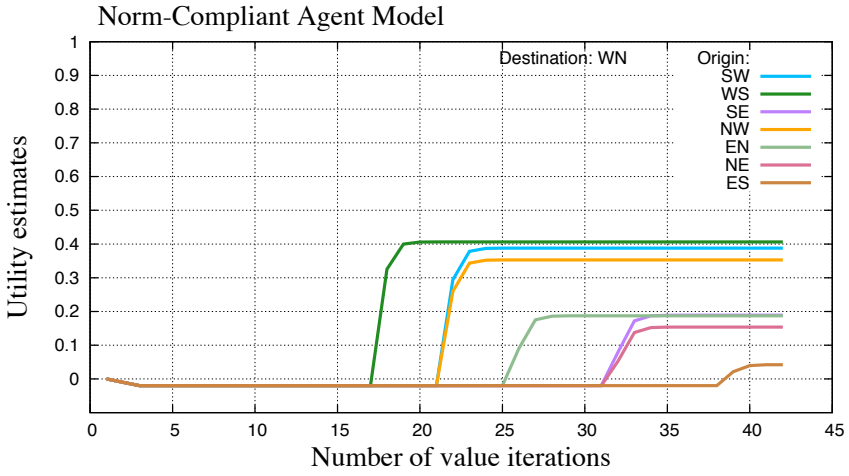
To construct an optimal policy with VI, both agent models had to perform 42 value iterations. Although the number of value iterations is the same for both agent models, the time spent by each value iteration in the self-interested model is significantly higher. As we can observe in Table 5.3, the construction of an optimal policy using VI in the norm-compliant agent model is ∼4.66 faster than in the self-interested agent model. The times shown in Table 5.3 have been measured with the JIP in millisecond (ms) precision.

Table 5.3: Agent models and their respective number of value iterations, and total time in milliseconds consumed in the construction of an optimal policy using the VI algorithm.

Agent model	Value iterations	Total time (ms)
Norm-compliant	42	2599,2
Self-interested	42	11816,9

Figure 5.21 shows the evolution of the utility estimates as the VI algorithm iterates: (a) shows the utilities for the norm-compliant agent heading to WN, while (b) shows the utilities for the self-interested agent heading to the same destination. Each line in the graph illustrates the evolution of the expected utility for running from a particular origin gateway. Starting with initial values of zero, the utilities immediately drop to -0.02 (two actions, starting and ending in the same gateway) until, at some point, a path is found to the destination, whereupon the utility rises.

(a)



(b)

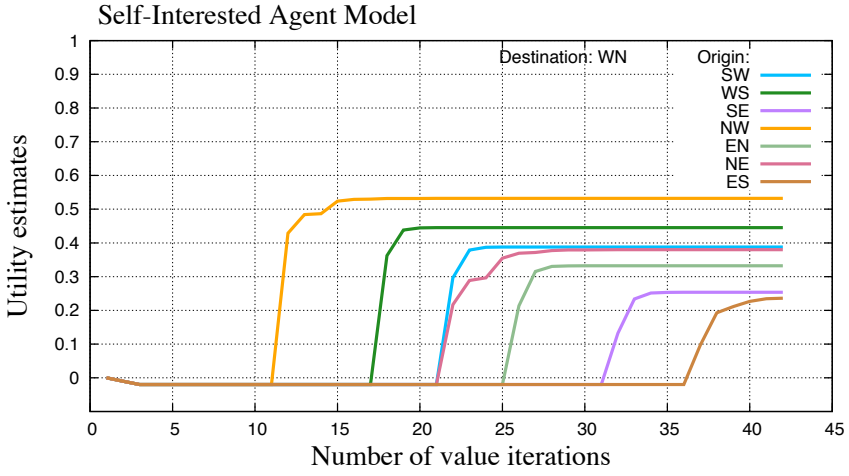


Figure 5.21: Graphs showing the evolution of the utilities as the VI algorithm iterates: (a) shows the utilities for the norm-compliant agent, while (b) shows the utilities for the self-interested agent. The destination gateway for both agents is WN. The lines illustrate the utility evolution for the agent from the possible origin gateways.

To construct an optimal policy, the MPI with order 5 required less time than the VI. The order of the MPI (number of value iterations per policy evaluation) has been selected on the basis of some preliminary experiments. The initial policy consists of executing MOVE where this action is admissible; in the states where this action is not available, the initial policy determines the execution of START, END, TURN or STOP, in this order, according to their availability.

Table 5.4 shows the time in milliseconds (ms) and the number of policy iterations to compute an optimal policy with the MPI. The time consumed by the self-interested model using the selected setup has been ∼2.4 times greater than the time consumed by the norm-compliant model. While the self-interested model made 9 policy iterations to compute an optimal policy, the norm-compliant model has found it with 7 policy iterations. Although the norm-compliant model performs more policy iterations, the time consumed by each iteration is significantly lower. This is explained by the fact that the state space of the norm-compliant agent is smaller, which saves time during the policy evaluation and policy improvement.

Table 5.4: Agent model (MPI order), number of policy iterations, and total time in milliseconds consumed in the construction of an optimal policy using the given MPI order.

Agent model (MPI order)	Policy iterations	Total time (ms)
Norm-compliant (5)	9	1601,9
Self-interested (5)	7	3836,0

In the experiments, the *norm-compliant* agent model overpassed the *self-interested* agent model according to criterion (c1), given that the norm-compliant consumed less computational resources than the self-interested during the reasoning processes.

5.6 Discussion

This chapter has presented a series of experiments in a simulated multiagent motion environment with the intention of measuring the impact of norms on the sequential decision-making of norm-aware agents. The experiments have been performed under a wide range of settings and the results have been analyzed and compared by way of well defined performance criteria.

5.6.1 Impact of norms on agents' utility

In the *single agent* experiments (e1), for any combination of penalty coefficient λ (s1) and enforcement intensity β (s2), the self-interested agents outperform or perform as well as the norm-compliant agents according to the *average utility* criterion (c2). Here, the coordination function of the norms is superfluous, however, the norm-compliant agents cannot detect this. The lower the penalty and detection probability, the more self-interested agents are willing to explore the forbidden partitions of the state space, which in single agent settings, lead to higher agent utility.

In the *multiagent* experiments with *homogeneous populations* (e2), when the number of agents exceeds a certain threshold (see Figure 5.15 (a)), the usefulness of norms as a coordination device becomes apparent. As an effect of the adequate coordination provided by the norms, norm-compliant agents have a larger *average utility* (c2) than self-interested agents. Thus, in this case, norms diminish the computational effort of decision making and help to achieve coordination in highly populated environments.

5.6.2 Norms and coordination problems

In the *multiagent* experiments (e2) and (e3), the largest the penalties λ (s1) and norm enforcement intensities β (s2), the lower is the number of crashes (c3). In Figure 5.14 we can observe that some combinations of large values for λ and β enforce full norm compliance and ensure the absence of crashes.

In the *multiagent* experiments with *heterogeneous populations* (e3), as we can see in Figure 5.18, the average percentage of agents that crash increases as the number of agents in the environment increases. If we increase the number of self-interested agents, the percentage of agents that crash rises at higher rates than increasing the number of norm-compliant agents.

5.6.3 Income from penalties against norm violations

From the macro perspective, the combinations of penalties λ (s1) and norm enforcement intensities β (s2) that make the self-interested agents comply with the norms result in no income from penalties (c4) as there are no norm violations to be penalized. With the rest of combinations of λ and β, the income tends to increase as λ and β rise, except for some small descents caused by reductions on the amount of norm violations committed by the agents.

In *multiagent* settings, the income from penalties against norm violations (c4) is proportional to the number of self-interested agents running in the motion simulator. For example, compare the income from a single self-interested agent, shown in Figure 5.11, with the income from 10 self-interested agents, shown in Figure 5.16. The last is approximately 10 times larger than the first.

5.6.4 Norm abidance and the cost of making rational decisions

The results presented in this chapter illustrate how norms can be used to prune the agents' search space. Thus, the reasoning problem that genuinely norm-compliance agents have to face becomes simpler (they only reason with the norm-compliant states). As a result, their *cognitive load* is reduced and it takes them less time to compute their policies, when compared to self-interested agents that need to search the entire problem space.

In the experiments, the *normative reasoning* in the norm-compliant agents is ∼1.35 times faster than in the self-interested agents, and the *size of state space* of the norm-compliant agents is 27% the size of the state space of the self-interested agents. Using Value Iteration (s4), the norm-compliant agents construct an optimal policy ∼4.66 times faster than the self-interested agents, and using Modified Policy Iteration (s4), the norm-compliant agents construct an optimal policy ∼2.4 times faster than their self-interested acquaintances.

Chapter 6

Evaluation of contracts

Any time you take a chance you better be sure the rewards are worth the risk because they can put you away just as fast for a ten dollar heist as they can for a million dollar job.

Stanley Kubrick

In regulated multiagent systems [52, 123, 32] the agents may be subject to mechanisms for adjusting their behaviour at some level in order to orchestrate a global behaviour. The usefulness of such regulations becomes more prominent in open systems, where heterogeneous agents are able to join and leave the system at runtime. In these open systems, there are no guarantees that the agents will act in a particular manner, and in this case, the establishment of some type of control over them makes possible the coordination of tasks and the avoidance of undesired states of the world.

Contracts have demonstrated to be suitable for regulating the behaviour of agents in several domains, such as for service procurement in the insurance industry, service level agreement management in software engineering, and aircraft engine aftercare. While we do not detail the specific applications in this thesis, Jakob et al. [62] provide substantial further details of the use cases for these. In particular, contracts express the responsibilities of each of the involved parties through the specification of norms; it is this core normative aspect of contracts that is the focus of this chapter. These contracts thus represent agreements of the parties, making explicit what each party can expect from the others, but providing flexibility in how they accomplish their own obligations.

Once the contracts have been specified by means of norms, the contracts' lifetime can be divided in two sequential phases:

- *Pre-signing:* In order to *evaluate* the contracts, the agents calculate the expected earnings of signing the contracts and the norm violations probabilities (risks) by the addressees of the contracts; then, based on the information obtained in the evaluations, the agents engage in *negotiations* [104, 99, 60, 12] that might result in agreements;

- *Post-signing:* If the contract is signed, the agents engage in its *execution*; the signed contract establishes a commitment whose violation implies the execution of sanctions; in order to detect contract violations, *monitoring* activities are performed [81, 82].

In order to behave rationally in a system regulated by contracts, a *self-interested* agent must be capable of calculating the expected utilities of signing them, and estimating the likelihood, or risk, of norm violations for a particular course of action. In this chapter, we focus on the *pre-signing* phase, aiming at the *evaluation* of contracts by rational agents in stochastic environments where norm violations may happen intentionally or as a consequence of the intrinsic uncertainty of the system.

The idea behind this chapter is to demonstrate, by way of a case study in a simulated aerospace aftermarket, the applicability of the NMDP framework to model contracts in stochastic sequential decision making settings. Furthermore, this case study demonstrates that the general algorithms introduced in Chapter 4 provide a principled method that can be used to determine contract violations, quantitatively evaluate contracts and assess contract violations risks.

The chapter is organized as follows. Section 6.1 introduces the aerospace aftermarket domain. Section 6.2 details the case study developed in this domain. Section 6.3 shows how an engine manufacturer agent can be modeled with the NMDP framework. Section 6.4 describes how contract violations can be identified and how the risks of these contract violations can be calculated in a stochastic aerospace aftermarket environment. Finally, Section 6.5 closes this chapter by promoting a discussion.

6.1 Aerospace aftermarket domain

The aerospace aftermarket use case described in this section was first introduced by Jakob et al. [62], and further developed by Meneguzzi et al. [78]. According to the authors, the aerospace aftermarket is increasingly populated by customers buying a service rather than a product. In this domain, the aircraft engine manufacturers provide long term commitments, which make these manufacturers responsible for providing serviceable engines to airline operators at specific locations (a given engine manufacturer may service an airline operator at multiple sites), not allowing any aircraft to be idle for greater than an agreed duration.

Minimum service level commitments are stipulated in *aftercare contracts*. If these commitments are violated (for example, when an airline aircraft is on the ground, awaiting functioning engines for a period of time greater than that agreed with the engine manufacturer), then engine manufacturers receive predetermined financial penalties. In this business model, servicing and maintenance becomes a key driver of profitability for the engine manufacturer since aftercare contracts are worth millions of euros.

On the basis of these aftercare contracts, the engine manufacturers establish *parts supply contracts* with engine part suppliers. In order to repair an engine, a manufacturer requests the required engine parts from contracted suppliers. Once all engine parts have been obtained, the manufacturer resumes repairing the engine, and readies it in the designated aircraft, notifying the airline operator that the repair has been accomplished.

There are three relevant types of autonomous agents in the aerospace aftermarket domain (see Figure 6.1):

- *Airline operators* are the customers for aftercare contracts; each operator has its own fleet of aircraft which needs to be kept in service;
- *Engine manufacturers* are providers of aftercare contracts; they attempt to perform the engine repair as specified in the contract or incur penalties; moreover, manufacturers are customers for parts supply contracts;
- *Part suppliers* are providers of parts supply contracts; they deliver engine parts to the manufacturers.

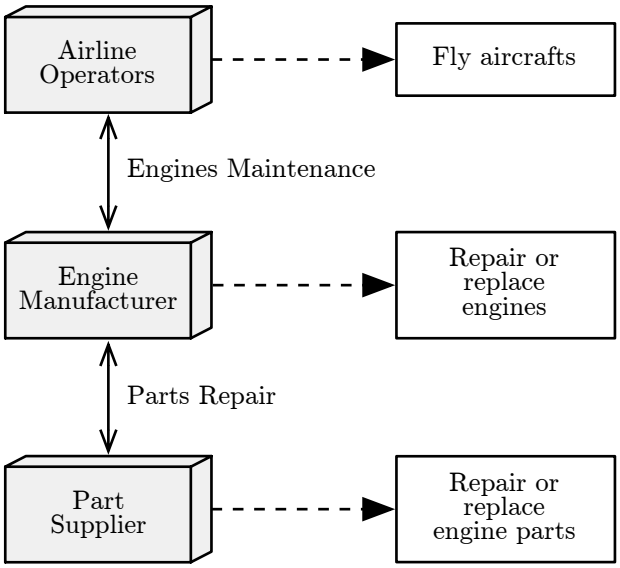


Figure 6.1: Actors and services in the aerospace aftermarket domain.

The provision of services is regulated by contractual agreements between the involved parties. Two particular types of contract are studied in this work:

- *Aftercare contract* which specifies the terms and conditions under which an engine manufacturer undertakes to supply and maintain engines for an airline operator; an aftercare contract can specify, for example, the serviceable engine rate, financial penalties applicable if the agreed service levels are not met, etc;
- *Parts supply contract* which regulates how an engine manufacturer asks a supplier of engine parts to produce and deliver new parts or refurbished old parts of a given type over a given period; this contract can specify, for example, the locations where part supplies should be delivered, the cost of parts, delivery times, etc.

6.2 Description of the case study

The present case study is develop in an aerospace aftermarket populated by an *engine manufacturer* (AGEM), an *airline operator* (AGAO) and three *engine part suppliers* (AGSX, AGSY and AGSZ). The case study is made from the perspective of the engine manufacturer, which is faced with deciding between contracts: an aftercare contract COAC to be signed with AGAO, and three parts supply contracts, COPSX, COPSY and COPSZ, which can be signed with AGSX, AGSY and AGSZ, respectively (for the sake of simplicity, it is assumed that a single part P is required to repair the engine, and the three part suppliers are capable of providing this part). Thus, the manufacturer AGEM signs one aftercare contract and one parts supply contract, which results in three possible sets of contracts:

$$\begin{aligned}\text{SETCO}_1&=\{\text{COAC, COPSX}\}\\ \text{SETCO}_2&=\{\text{COAC, COPSY}\}\\ \text{SETCO}_3&=\{\text{COAC, COPSZ}\}\end{aligned}$$

The case study developed in this chapter focuses on the engine manufacturer agent AGEM, showing in Section 6.3 how this agent can be modeled with the NMDP framework introduced in Chapter 3, and describing in Section 6.4 how the self-interested agent model introduced in Chapter 4 can be employed to reason about contracts.

6.3 Engine manufacturer agent

This section describes the NMDP instances, which represent the knowledge of the *engine manufacturer agent* (AGEM) inhabiting the simulated aftermarket introduced in Section 6.2. There are three NMDP instances, one for each set of contracts:

$$\begin{aligned}nmdp_1 &= \langle \mathcal{S}_1, \mathcal{A}, \mathcal{C}_1, \mathcal{T}_1, \mathcal{R}_1, \mathcal{N}_1 \rangle\\ nmdp_2 &= \langle \mathcal{S}_2, \mathcal{A}, \mathcal{C}_2, \mathcal{T}_2, \mathcal{R}_2, \mathcal{N}_2 \rangle\\ nmdp_3 &= \langle \mathcal{S}_3, \mathcal{A}, \mathcal{C}_3, \mathcal{T}_3, \mathcal{R}_3, \mathcal{N}_3 \rangle\end{aligned}$$

The state space of the NMDPs is described using sets of multi-valued features:

- *Engine part* (f_1). Has AGEM ordered the part P required to perform the repair of the engine? If false, this feature assumes the value $\overline{\text{O}}$. If the engine part P has been ordered, it assumes the value O. If P has been received, the feature assumes the value X, Y or Z, which corresponds to the respective supplier AGSX, AGSY or AGSZ.

- *Order deadline* (f_2). Is the engine part supplied by the order deadline specified in the supply contract? Initially, this feature assumes the value A, which means that the supplier still have time to deliver the part P; if this deadline has passed and the part has not been received, this feature assumes the value $\overline{\text{A}}$, otherwise it remains A.

- *Engine condition* (f_3). Is the engine repaired by engine manufacturer AGEM? Is the engine handed over to the airline operator AGAO? If the engine is not repaired, then this feature assumes the value $\overline{\text{R}}$; if the engine is repaired but not delivered, it assumes the value R; if the engine is repaired and delivered, it assumes the value D.

- *Repair deadline* (f_4). Has AGEM repaired the engine by the deadline in the aftercare contract COAC with AGAO? Initially, this feature assumes the value B, which means that AGEM have time to repair and deliver the engine; if this deadline has passed and the engine has not been repaired and delivered, it assumes the value $\overline{\text{B}}$.

For instance, the initial state $\overline{\text{O}}\text{A}\overline{\text{R}}\text{B}$ means the part P has not been ordered, the engine is neither repaired nor delivered, and the engine manufacturer AGEM has satisfied both deadlines. In this work, deadlines are represented simply as features of the state space. A more sophisticated treatment of time in normative systems, such as the use of branching time logic [25] is certainly interesting, but not necessary for the purpose of this case study.

The action space is composed of six actions: order an engine part from a supplier, receive an ordered part, repair an engine and deliver an engine.

$$\begin{aligned}\mathcal{A} = \{\ & \text{order(Part, Supplier)},\\ & \text{receive(Part, Supplier)},\\ & \text{repair(Engine)},\\ & \text{deliver(Engine, Operator)}\ \}\end{aligned}$$

For example, Figure 6.2 illustrates the state space of the NMDP instance $nmdp_1$ and the admissible actions in each state (capabilities $\mathcal{C}_1$). The label i indicates the initial state of the process, while the label t indicates the absorbing (terminal) states. In $nmdp_2$, the feature X is rseplaced by Y and AGSX is replaced by AGSY, while in $nmdp_3$, Z replaces X and AGSZ replaces AGSX.

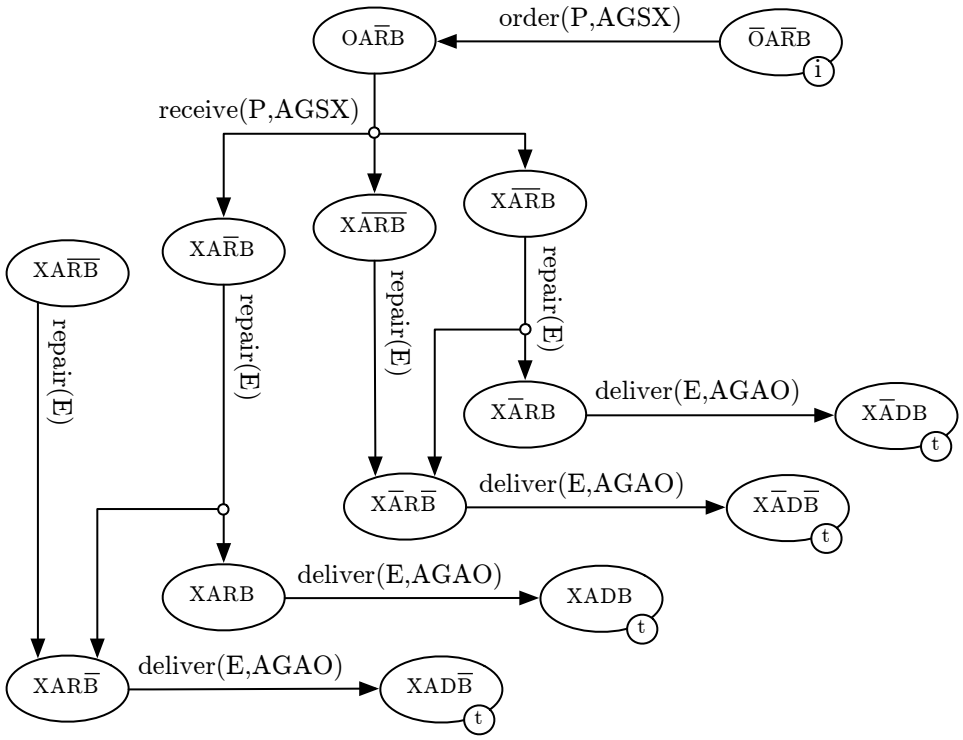


Figure 6.2: State space of $nmdp_1$ and admissible actions in each state ($\mathcal{C}_1$).

In this simulated aerospace aftermaket, *order* and *deliver* are deterministic, but *receive* and *repair* are not. The state-transitions caused by these non-deterministic actions are specified as follows:

$$\begin{aligned}\mathcal{T}_1 = \{\ &(\text{OA}\overline{\text{R}}\text{B},\ \text{receive(P,AGSX)},\ \text{XA}\overline{\text{R}}\text{B}) \rightarrow 0.85,\\ &(\text{OA}\overline{\text{R}}\text{B},\ \text{receive(P,AGSX)},\ \text{X}\overline{\text{AR}}\text{B}) \rightarrow 0.10,\\ &(\text{OA}\overline{\text{R}}\text{B},\ \text{receive(P,AGSX)},\ \text{X}\overline{\text{AR}}\overline{\text{B}}) \rightarrow 0.05,\\ &(\text{XA}\overline{\text{R}}\text{B},\ \text{repair(E)},\ \text{XARB}) \rightarrow 0.95,\\ &(\text{XA}\overline{\text{R}}\text{B},\ \text{repair(E)},\ \text{XAR}\overline{\text{B}}) \rightarrow 0.05,\\ &(\text{X}\overline{\text{AR}}\text{B},\ \text{repair(E)},\ \text{X}\overline{\text{A}}\text{RB}) \rightarrow 0.95,\\ &(\text{X}\overline{\text{AR}}\text{B},\ \text{repair(E)},\ \text{X}\overline{\text{A}}\text{R}\overline{\text{B}}) \rightarrow 0.05,\ \ldots\ \}\end{aligned}$$

$$\begin{aligned}\mathcal{T}_2 = \{\ &(\text{OA}\overline{\text{R}}\text{B},\ \text{receive(P,AGSY)},\ \text{YA}\overline{\text{R}}\text{B}) \rightarrow 0.90,\\ &(\text{OA}\overline{\text{R}}\text{B},\ \text{receive(P,AGSY)},\ \text{Y}\overline{\text{ARB}}) \rightarrow 0.09,\\ &(\text{OA}\overline{\text{R}}\text{B},\ \text{receive(P,AGSY)},\ \text{Y}\overline{\text{ARB}}) \rightarrow 0.01,\\ &(\text{YA}\overline{\text{R}}\text{B},\ \text{repair(E)},\ \text{YARB}) \rightarrow 0.95,\\ &(\text{YA}\overline{\text{R}}\text{B},\ \text{repair(E)},\ \text{YAR}\overline{\text{B}}) \rightarrow 0.05,\\ &(\text{Y}\overline{\text{ARB}},\ \text{repair(E)},\ \text{Y}\overline{\text{A}}\text{RB}) \rightarrow 0.95,\\ &(\text{Y}\overline{\text{ARB}},\ \text{repair(E)},\ \text{Y}\overline{\text{A}}\text{R}\overline{\text{B}}) \rightarrow 0.05,\ \ldots\ \}\end{aligned}$$

$$\begin{aligned}\mathcal{T}_3 = \{\ &(\text{OA}\overline{\text{R}}\text{B},\ \text{receive(P,AGSZ)},\ \text{ZA}\overline{\text{R}}\text{B}) \rightarrow 0.98,\\ &(\text{OA}\overline{\text{R}}\text{B},\ \text{receive(P,AGSZ)},\ \text{Z}\overline{\text{ARB}}) \rightarrow 0.01,\\ &(\text{OA}\overline{\text{R}}\text{B},\ \text{receive(P,AGSZ)},\ \text{Z}\overline{\text{ARB}}) \rightarrow 0.01,\\ &(\text{ZA}\overline{\text{R}}\text{B},\ \text{repair(E)},\ \text{ZARB}) \rightarrow 0.95,\\ &(\text{ZA}\overline{\text{R}}\text{B},\ \text{repair(E)},\ \text{ZAR}\overline{\text{B}}) \rightarrow 0.05,\\ &(\text{Z}\overline{\text{ARB}},\ \text{repair(E)},\ \text{Z}\overline{\text{A}}\text{RB}) \rightarrow 0.95,\\ &(\text{Z}\overline{\text{ARB}},\ \text{repair(E)},\ \text{Z}\overline{\text{A}}\text{R}\overline{\text{B}}) \rightarrow 0.05,\ \ldots\ \}\end{aligned}$$

According to the transition model, *receive* changes the engine part feature from O to X, Y or Z, depending on the supplier; if the engine part is not received from the supplier by the deadline, then the order deadline feature changes from A to $\overline{\text{A}}$. The action *repair* changes the repair feature from $\overline{\text{R}}$ to R; if the repair is not performed by the deadline, the repair deadline feature changes from B to $\overline{\text{B}}$.

Dealing with different counterparts may provide different rewards, so each NMDP instance has its own reward function. $\mathcal{R}_2$ and $\mathcal{R}_3$ are partially specified as the rest of their entries are equal to $\mathcal{R}_1$ except for X that is replaced by Y and Z, respectively.

$$\begin{aligned}\mathcal{R}_1 = \{&(\overline{\text{O}}\text{A}\overline{\text{R}}\text{B},\ \text{order(P,AGSX)},\ \text{OA}\overline{\text{R}}\text{B}) \rightarrow -1.0,\\ &(\text{OA}\overline{\text{R}}\text{B},\ \text{receive(P,AGSX)},\ \text{XA}\overline{\text{R}}\text{B}) \rightarrow 0.0,\\ &(\text{OA}\overline{\text{R}}\text{B},\ \text{receive(P,AGSX)},\ \text{X}\overline{\text{ARB}}) \rightarrow 0.0,\\ &(\text{OA}\overline{\text{R}}\text{B},\ \text{receive(P,AGSX)},\ \text{X}\overline{\text{ARB}}) \rightarrow 0.0,\\ &(\text{XA}\overline{\text{R}}\text{B},\ \text{repair(E)},\ \text{XAR}\overline{\text{B}}) \rightarrow -1.0,\\ &(\text{XA}\overline{\text{R}}\text{B},\ \text{repair(E)},\ \text{XAR}\overline{\text{B}}) \rightarrow -1.0,\\ &(\text{XA}\overline{\text{R}}\text{B},\ \text{repair(E)},\ \text{XARB}) \rightarrow -1.0,\\ &(\text{X}\overline{\text{ARB}},\ \text{repair(E)},\ \text{X}\overline{\text{A}}\text{R}\overline{\text{B}}) \rightarrow -1.0,\\ &(\text{X}\overline{\text{ARB}},\ \text{repair(E)},\ \text{X}\overline{\text{A}}\text{R}\overline{\text{B}}) \rightarrow -1.0,\\ &(\text{X}\overline{\text{ARB}},\ \text{repair(E)},\ \text{X}\overline{\text{A}}\text{RB}) \rightarrow -1.0,\\ &(\text{XAR}\overline{\text{B}},\ \text{deliver(E,AGAO)},\ \text{XAD}\overline{\text{B}}) \rightarrow 3.0,\\ &(\text{XARB},\ \text{deliver(E,AGAO)},\ \text{XADB}) \rightarrow 3.0,\\ &(\text{X}\overline{\text{A}}\text{R}\overline{\text{B}},\ \text{deliver(E,AGAO)},\ \text{X}\overline{\text{A}}\text{D}\overline{\text{B}}) \rightarrow 3.0,\\ &(\text{X}\overline{\text{A}}\text{RB},\ \text{deliver(E,AGAO)},\ \text{X}\overline{\text{A}}\text{DB}) \rightarrow 3.0\ \}\end{aligned}$$

$$\begin{aligned}\mathcal{R}_2 = \{&(\overline{\text{O}}\text{A}\overline{\text{R}}\text{B},\ \text{order(P,AGSY)},\ \text{O}\text{A}\overline{\text{R}}\text{B}) \rightarrow -0.8,\\ &(\text{O}\text{A}\overline{\text{R}}\text{B},\ \text{receive(P,AGSY)},\ \text{Y}\text{A}\overline{\text{R}}\text{B}) \rightarrow 0.0,\\ &(\text{O}\text{A}\overline{\text{R}}\text{B},\ \text{receive(P,AGSY)},\ \text{Y}\overline{\text{ARB}}) \rightarrow 0.0,\\ &(\text{O}\text{A}\overline{\text{R}}\text{B},\ \text{receive(P,AGSY)},\ \text{Y}\overline{\text{AR}}\text{B}) \rightarrow 0.0,\ \dots\ \}\end{aligned}$$

$$\begin{aligned}\mathcal{R}_3 = \{&(\overline{\text{O}}\text{A}\overline{\text{R}}\text{B},\ \text{order(P,AGSZ)},\ \text{O}\text{A}\overline{\text{R}}\text{B}) \rightarrow -1.5,\\ &(\text{O}\text{A}\overline{\text{R}}\text{B},\ \text{receive(P,AGSZ)},\ \text{Z}\text{A}\overline{\text{R}}\text{B}) \rightarrow 0.0,\\ &(\text{O}\text{A}\overline{\text{R}}\text{B},\ \text{receive(P,AGSZ)},\ \text{Z}\overline{\text{ARB}}) \rightarrow 0.0,\\ &(\text{O}\text{A}\overline{\text{R}}\text{B},\ \text{receive(P,AGSZ)},\ \text{Z}\overline{\text{AR}}\text{B}) \rightarrow 0.0,\ \dots\ \}\end{aligned}$$

The first contract is an aftercare contract between the engine manufacturer AGEM and the airline operator AGAO. This contract, named COAC, includes the following norms:

- q_1 – An obligation on the engine manufacturer AGEM to repair and deliver the engine by the repair deadline B; if this norm is violated then the engine manufacturer will have to pay a penalty to the airline operator AGAO in order to accomplish the delivery (paying the penalty resets the repair deadline); the penalty is -2.0; any transition that arrives at states with the repair deadline feature equal to $\overline{\text{B}}$ characterizes a violation.

- q_2 – A prohibition on the manufacturer AGEM to repair the engine with parts from AGSY; if the manufacturer repairs the engine with parts from AGSY then the airline operator AGAO refuses the engine and charges a penalty of -5.0; this norm is violated if the manufacturer achieves any state with the features Y and R.

In the supply contracts, the engine manufacturer AGEM is not an addressee of the norms since the obligations are on the suppliers. However, the manufacturer, as the beneficiary of the contract, expects to receive the engine parts by the deadline. The following three norms correspond to the parts supply contracts COPSX, COPSY and COPSZ, with AGSX, AGSY and AGSZ, respectively:

- q_3 – An obligation on the supplier AGSX to produce and deliver the requested part P by the order deadline A; if this norm is violated, the engine manufacturer can charge a penalty on the part supplier AGSX for delaying the delivery (charging resets the order deadline); the reward for charging the penalty is 1.0; any transition that arrives at states with the order deadline feature equal to $\overline{\text{A}}$ characterizes a violation of this norm.

- q_4 – As in q_3 except that AGSY replaces AGSX.
- q_5 – As in q_3 except that AGSZ replaces AGSX and the reward for charging the penalty is 1.5.

So, for each set SETCO_i we construct a set of norms $\mathcal{N}_i$ containing the union of the contractual norms in SETCO_i:

$$\mathcal{N}_1 = \text{COAC} \bigcup \text{COPSX} = \{q_1, q_2, q_3\}$$
$$\mathcal{N}_2 = \text{COAC} \bigcup \text{COPSY} = \{q_1, q_2, q_4\}$$
$$\mathcal{N}_3 = \text{COAC} \bigcup \text{COPSZ} = \{q_1, q_2, q_5\}$$

The norms in the contracts are encoded according to Definition 5 (see Chapter 3). In order to represent the *beneficiaries* of the norms, each element in $\mathcal{G}$ is defined as follows:

$$\textsc{Addressee} \rightarrow \textsc{Beneficiary}$$

where Addressee is the agent to which the norm applies, and Beneficiary is the agent that benefits from the norm compliance, to which the addressee pays the penalty in case of norm violation.

The specification of the norms is as follows:

$$\begin{aligned}
q_1 = \langle & 1, \textsc{Obligation}, \\
& \{\text{AGEM} \rightarrow \text{AGAO}\}, \\
& \top, (f_4{=}\textsc{b}), \\
& \quad \langle \{ \top \rightarrow -2.0 \}, \\
& \quad \{ \top \rightarrow \{(f_4{=}\textsc{b})\} \} \rangle \rangle \\
q_2 = \langle & 2, \textsc{Prohibition}, \\
& \{\text{AGEM} \rightarrow \text{AGAO}\}, \\
& (f_3{=}\textsc{r}) \vee (f_3{=}\textsc{d}), (f_1{=}\textsc{y}), \\
& \quad \langle \{ \top \rightarrow -5.0 \}, \\
& \quad \{ \top \rightarrow \{(f_3{=}\textsc{r})\} \} \rangle \rangle \\
q_3 = \langle & 3, \textsc{Obligation}, \\
& \{\text{AGSX} \rightarrow \text{AGEM}\}, \\
& \neg(f_1{=}\overline{\textsc{o}}), (f_2{=}\textsc{a}), \\
& \quad \langle \{ \top \rightarrow +1.0 \}, \\
& \quad \{ \top \rightarrow \{(f_2{=}\textsc{a})\} \} \rangle \rangle
\end{aligned}$$

$$
\begin{aligned}
q_4 = \langle 4,\ &\textsc{Obligation},\\
&\{\text{AGSY} \rightarrow \text{AGEM}\},\\
&\neg(f_1{=}\overline{\text{O}}),\ (f_2{=}\text{A}),\\
&\qquad \langle\ \{\ \top \rightarrow +1.0\ \},\\
&\qquad\quad \{\ \top \rightarrow \{(f_2{=}\text{A})\}\ \}\ \rangle\rangle\\
q_5 = \langle 5,\ &\textsc{Obligation},\\
&\{\text{AGSZ} \rightarrow \text{AGEM}\},\\
&\neg(f_1{=}\overline{\text{O}}),\ (f_2{=}\text{A}),\\
&\qquad \langle\ \{\ \top \rightarrow +1.5\ \},\\
&\qquad\quad \{\ \top \rightarrow \{(f_2{=}\text{A})\}\ \}\ \rangle\rangle
\end{aligned}
$$

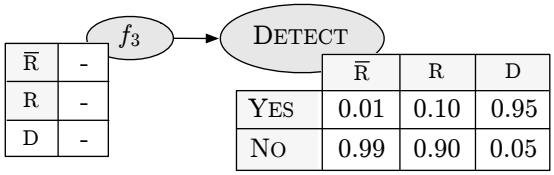
Notice that in q_1 and q_2, AGEM is the addressee of the norms and the penalty is a *negative* value to be paid by this agent in case of norm violation. In q_3, q_4 and q_5, AGEM is the beneficiary, and for this reason the penalty is a *positive* value, meaning that AGEM receives this reward as a compensation for the norm violation performed by the part supplier.

The detection function of AGEM is defined by means of Bayesian networks, an example of compact representation introduced in Section 3.3.3. The conditional probabilities in the networks have been arbitrarily chosen. Figure 6.3 shows the Bayesian networks, explained as follows:

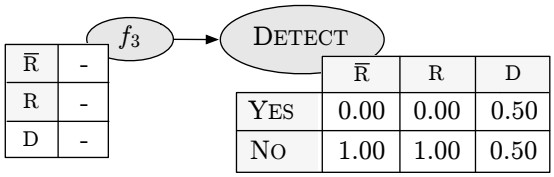
(a) The norm q_1 is violated if the delivery deadline passes ($f_4 = \overline{\text{B}}$), and the detection of its violations (and imposition of sanction) is more likely to happen when the engine has been delivered ($f_3 = \overline{\text{D}}$). For this reason, the detection probabilities for q_1 depend only on the engine condition (f_3).

(b) The violation of q_2 can only be detected when the engine has been delivered to AGON (an inspection is necessary to check the supplier of the part used in the maintenance). As in q_1 the detection probabilities for q_2 depend only on f_3.

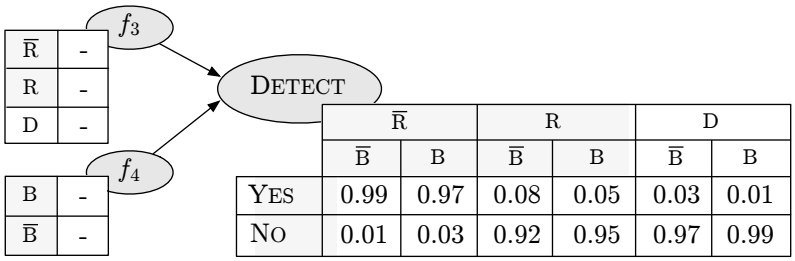
(c) The norms q_3, q_4 and q_5 are violated if the order deadline passes ($f_2 = \overline{\text{A}}$). The violation of these norms is more likely to be detected when the engine has not been repaired ($f_3 = \overline{\text{R}}$), in the first stages of the process when AGEM is waiting for the part. Furthermore, it is more likely to detect that the order deadline has passed if the delivery deadline has passed as well ($f_4 = \overline{\text{B}}$).



(a) Bayesian network representing the detection probabilities for q_1



(b) Bayesian network representing the detection probabilities for q_2



(c) Bayesian network representing the detection probabilities for q_3, q_4 and q_5

Figure 6.3: Detection function of AGEM defined with Bayesian networks.

6.4 Reasoning about contracts

6.4.1 Identifying norm violations and representing sanctions

In this section, the *self-interested* agent model, introduced in Section 4.2, is used to implement the engine manufacturer agent. In this case study, there is no policy that ensures the absence of norm violations, and for this reason, the *norm-compliant* agent model is not applicable.

First of all, the normative reasoning is performed with the NMDP instances. Algorithm 3 is called to identify which states violate which contractual norms, and Algorithm 4 is called to represent the respective sanctions with the norm violating states. Thus, for each $nmdp_i$ described in Section 6.3, there will be a mdp_i.

Figure 6.4 illustrates mdp_1, the MDP resulting from the *normative reasoning* over $nmdp_1$ using the *self-interested* agent model. The violating states are drawn as red nodes with labels indicating which norms are violated in these states. From each violating state, new state-transitions have been added to represent the sanctions. The state s_ι and the action a_ι, used to represent sanctions from absorbing states, are added as well.

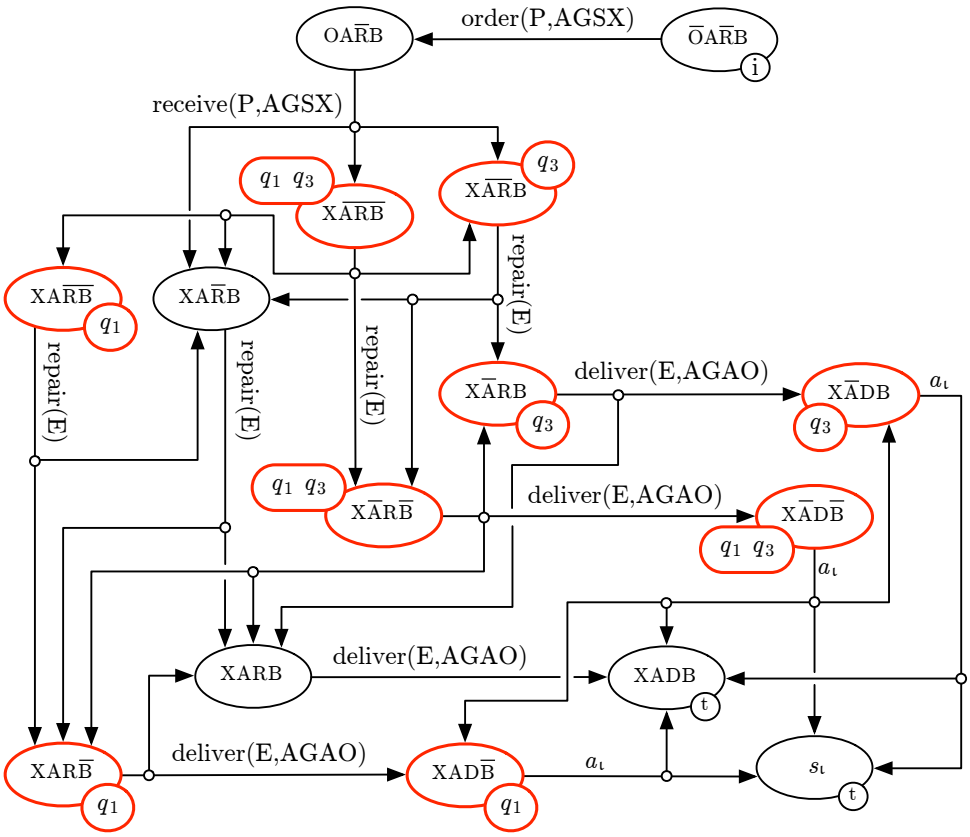


Figure 6.4: MDP resulting from the *normative reasoning* over $nmdp_1$ using the *self-interested* agent model. The norm violating states are represented as red nodes with labels indicating which norms are violated in the respective states.

For example, consider $\text{X}\overline{\text{ARB}}$ in Figure 6.4, which violates q_1 and q_3. If AGEM executes the action *repair* in this state, there are four possible outcomes:

1. $\text{X}\overline{\text{A}}\text{R}\overline{\text{B}}$ – If the process move to this state, no norm violation has been detected in the origin state $\text{X}\overline{\text{ARB}}$.

2. $\text{XA}\overline{\text{RB}}$ – If the process move from the origin state $\text{X}\overline{\text{ARB}}$ to this state, the violation of the norm q_3 has been detected, AGSX pays 1.0 to AGEM as a compensation for the norm violation (penalty) and the order deadline is reset (A).

3. $\text{X}\overline{\text{AR}}\text{B}$ – If the process move to this state, the violation of the norm q_1 has been detected, AGEM pays 2.0 to AGAO (penalty for not delivering the engine by the repair deadline) and the repair deadline is reset (B).

4. $\text{XA}\overline{\text{R}}\text{B}$ – If the process move from the origin state $\text{X}\overline{\text{ARB}}$ to this state, the violations of q_1 and q_3 have been detected. In this case, the agent receives 1.0 from AGSX and pays 2.0 to AGAO (AGEM loses 1.0). Both deadlines are reset.

6.4.2 Calculating expected utilities and risks

The decision taken by AGEM concerning which contracts to sign is supported by the expected utilities calculated for each resulting MDP using the Value Iteration algorithm [6]. The agent compares the expected utility (EU) for the initial state of each mdp_i, and then, it chooses the one with the maximum value. In this aerospace aftermarket case study, the engine manufacturer decides to sign the contracts in SETCO_1 given that mdp_1 provides the highest EU in $\overline{\text{O}}\text{A}\overline{\text{R}}\text{B}$:

$$\begin{aligned}
\text{EU}(\overline{\text{O}}\text{A}\overline{\text{R}}\text{B},\ mdp_1) &= 0.9603 \\
\text{EU}(\overline{\text{O}}\text{A}\overline{\text{R}}\text{B},\ mdp_2) &= -0.8708 \\
\text{EU}(\overline{\text{O}}\text{A}\overline{\text{R}}\text{B},\ mdp_3) &= 0.4158
\end{aligned}$$

In this case study, SETCO_2 is not optimal primarily because the violation of q_2 is very likely to be detected and it implies in a high penalty, and SETCO_3 is not optimal because the cost of purchasing the part P from AGSZ is more expensive than purchasing from AGSX.

At the moment that the states that violate the norms in the contracts in SETCO_i have been determined, the risks (probabilities) of violating a given norm q_j can be assessed. To do so, the following steps are taken:

(1) Each mdp_i is combined with its respective policy, obtaining this way a Markov Chain [66] referred to as mc_i.

(2) Every state of mc_i that violate q_j is transformed into an absorbing state by removing the arcs departing from it.

(3) Finally, the absorption probabilities in mc_i starting from the initial state $\overline{\text{O}}\text{A}\overline{\text{R}}\text{B}$ are calculated[1]. By summing the chances of ending in the absorbing states that violate q_j, we calculate the risk of violating q_j.

For example, Figure 6.5 and Figure 6.6 show the probabilities of violating the contracts in SETCO_1, which are COAC=$\{q_1, q_2\}$ and COPSX=$\{q_3\}$. The norm violating states are represented as red nodes labeled with the probability of reaching them from the initial state. In Figure 6.5 we can see the probabilities of violating q_1 which sum 0.09755, that is, 9.755%. The nodes reachable from XARB and X$\overline{\text{A}}$RB have been omitted as they do not violate q_1.

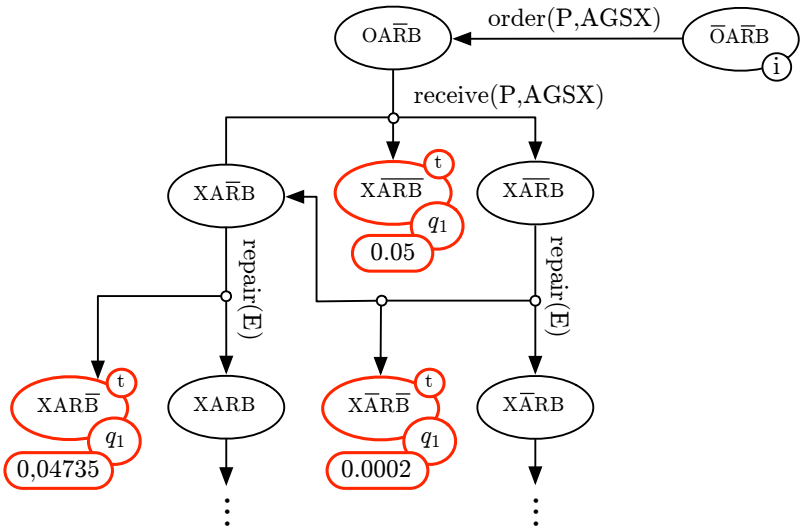


Figure 6.5: Probabilities of violating the norm q_1.

[1]The probability that an absorbing chain will be absorbed in a given absorbing state if it starts in a given transient state can easily calculated with the fundamental matrix and transition matrix in the canonical form. For details on the computation of the absorption probabilities, see [85].

In Figure 6.6 we can observe the probabilities of violating q_3 which sum 0.15. The norm q_2 cannot be violated in this example since AGEM does not purchase the engine part from AGSY. The nodes reachable from $\text{XA}\overline{\text{R}}\text{B}$ have been omitted as they do not violate the norm q_3.

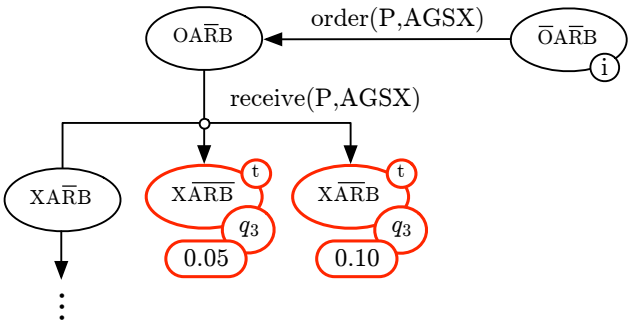


Figure 6.6: Probabilities of violating the norm q_3.

6.5 Discussion

This chapter has shown a case study in a simulated aerospace aftermarket domain, which demonstrates the applicability of the NMDP framework and self-interested agent model for evaluating contracts in sequential decision making settings. This case study shows how an engine manufacturer agent and a set of contracts can be modeled with the NMDP framework, and how this agent can evaluate these contracts by assessing their expected utilities and likelihoods, or risks, of norm violations for a particular course of action.

The related work in contractual agent societies is quite extensive and has covered a wide range of issues, such as frameworks, languages, negotiation protocols and techniques, monitoring mechanisms. Dellarocas [39] proposes an abstract model of open information systems where autonomous agents configure themselves through a set of dynamically negotiated contracts. Sallé [103] develop a framework for the automation of the lifecycle of contractual relationships between agents in a B2B platform. Kollingbaum and Norman [67] propose an approach, named supervised interaction, to the problem of establishing trust between contracting autonomous agents. The supervised interactions are made via an organizational framework, a contract specification

language and a contract management process. Governatori and Milosevic [56] formalize a system to provide a logic-based foundation for the policy aspects of the language BCL [74], developed to support business contract specification for contract monitoring. Oren et al. [88] formalize an electronic contracting language, in which clauses within contracts are specified as permissions, obligations and prohibitions on contract parties. The norms in this language are associated with a status, which allows for the determination of whether contract violations takes place and who is the responsible for causing the violations. Boella and van der Torre [12] formalize contracts as systems of regulative and constitutive norms, and develop a game theory wherein agents negotiate contracts in organizations. Modgil et al. [82] develop a global monitoring architecture for determining the source of violations in signed contracts. The monitor agents gather information from entrusted observers and generate explanations for violations within a supply chain in a simulated aerospace aftermarket.

In this context, the NMDP framework provides a language to specify not only contracts, but also the stochastic domains to be regulated by them. Even though the case study presented in this chapter does not include contract negotiation or contract monitoring, we believe that the type of evaluations shown here can be helpful to support decisions in negotiations and to improve monitoring activities by indicating situations in which contract violations are more likely to happen.

Part IV

Conclusion

Chapter 7

Conclusion

Science never solves a problem without creating ten more.

George Bernard Shaw

7.1 Contributions

The main contributions of this thesis are detailed as follows:

(i) *Generalization of MDPs to allow normative reasoning.*
In Chapter 3 we have focused on the problem of representing the knowledge of norm-aware rational agents. To this end, we have introduced the NMDP framework, an extension of the well-known MDP framework to allow the representation of normative structures and probabilistic knowledge regarding the detection of norm violations. The NMDP framework provides a principled method to check the norm compliance of states and the consistency of norms and sanctions (conflict detection). Furthermore, we have shown how factored state spaces can be exploited in order to compactly encode norms and detection models.

In a broader context, we believe that the NMDP framework can serve to ground new research on normative reasoning, in particular, the development of general algorithms for reasoning about norms in stochastic domains.

(ii) *Computational models of normative reasoning.*

To address the problem of reasoning about norms with the NMDP framework introduced in Chapter 3, two decision-theoretical models of normative reasoning have been developed in Chapter 4. In particular, these normative reasoning processes are specified as sound algorithms and embedded in two norm-aware agent models: self-interested agents incorporate sanctions into the state-transition and reward functions, while norm-compliant agents utilize norms to create smaller norm-compliant MDPs.

In these agent models, we cover all the properties employed in the analysis made in Section 2.5: norm-autonomy, explicitness of norms and sanctions, quantitative decision model and probabilistic planning. By taking advantage of the *synergy* between these properties, we enable the development of norm-aware agents capable of planning sequential actions for maximizing utilities in stochastic environments regulated by explicitly represented norms.

(iii) *Synergistic interactions between MDPs and norms.*

One of the key contributions of this thesis is that we explore the combination of native strengths of the MDPs and normative systems to cover their individual limitations. On the one hand, we use the analytic advantages of MDPs to model the domain of interaction and agent decision-making under uncertainty, and, on the other hand, we use norms to support computational leverage techniques and coordinate agents.

(a) *Quantitative evaluation of norms.*

From a micro perspective, our agent models allow the agents to calculate the impact of particular sets of norms on their expected utilities, taking into account uncertainty, rewards and sanctions. Such quantitative evaluation is especially vital in domains where the performance of the agents is linked to important metrics such as amount of crashes in a motion environment or profit in activities regulated by contracts. From a macro perspective, these quantitative evaluations have been made with the motion simulator, which keeps record of crashes and payments of penalties received from the self-interested agents. Several examples of quantitative evaluations of norms have been provided in Section 5.5. There, we observe the impact of various normative settings on the agents' individual behaviour and utility.

(b) *Norms to achieve computational leverage with MDP-based agents.*

By means of the norm-compliant agent model, we develop a novel approach to get computational leverage in the policy construction. Our technique restricts the state space to norm-compliant partitions, which results in gains in tractability by curtailing MDP policy search. Section 5.5.4 presents results where we can see significant speedups in MDP policy search due to the techniques introduced in this thesis.

(c) *Coordination of MDP-based agents.*

An MDP is a model of an individual agent unaware of other agents when running in multiagent systems, such as the motion simulator described in Section 5.3. In this context, the activities of the agents can interfere with each other, and this might have undesirable consequences. Assuming that fully norm-compliance ensures the absence of coordination problems, it is possible to achieve coordination in multiagent systems composed of norm-compliant agents and/or self-interested agents. The difference is that with norm-compliant agents, the coordination is unconditionally supported, and with self-interested agents, the coordination must be enforced via adequate sanctions and norm enforcement intensities.

Notice that the proposed coordination approach is flexible in the sense that changes in the norms do not require the reconstruction of the agents. Once the new norms are mirrored into the NMDP instance, the new policy can be automatically generated by the reasoning processes.

(iv) *Applications.*

To validate our approach, two applications have been developed in this thesis. In Chapter 5, we have measured the performance of the different agent types in relation to specific controlled settings in a simulated motion environment. These experiments illustrate the kind of quantitative evaluations that we envision in normative multiagent systems. Chapter 6 demonstrates, by way of a case study in a simulated aerospace aftermarket, the applicability of our approach to model and reason about contracts in stochastic sequential decision making settings.

7.2 Limitations

The thesis has unleashed the synergistic potential of combining MDPs and norm-aware agent models. By doing so, it has overcome some of the drawbacks of existing approaches. While the objectives mentioned in the introduction of this thesis have been fully achieved, some problems still pertain:

(i) Some limitations have been inherited from the MDP framework, which for instance accounts neither for partial observability, nor for joint actions.

(ii) In this thesis, the norms encoded in the NMDPs are assumed to be consistent, that is, there is no state that simultaneously complies with one norm and violates other. To face this limitation, we would start looking into the mechanisms for the resolution of normative conflicts proposed by Vasconcelos et al. [116].

(iii) In some applications, specially when modeling contracts, it might be interesting to have a more sophisticated treatment of time (e.g. deadlines). In this work, we simply represent deadlines as features of the state space that change their values when the respective deadlines pass. The work by Broersen et al. [25, 11] in normative reasoning with deadlines can be used as starting point for tackling this second limitation.

7.3 Future research directions

This thesis has opened several possible new areas for further research. This section summarizes the four main agenda for future research.

(i) To deal with problems in which the agent cannot directly observe the underlying state, we would like to put forward the *Partially Observable Normative Markov Decision Process framework* (PONMDP), an extension of the NMDP framework including a set of observations and a set of conditional observation probabilities. In addition, we intend to approach the specification of transitions and rewards using factored representations (in this thesis, we only use factored state spaces and factored detection models), and then, to revisit the algorithms proposed in Chapter 4 in order to deal with these compact representations.

(ii) In Chapter 5 we have seen that in certain situations the norm-compliant agents outperform the self-interested agents, and in other situations we have seen the opposite. Bearing in mind this fact, we would like to develop a more sophisticate norm-aware agent model that considers only a subset of norms for violation. By doing this, we can save some computational resources by partially following the norms, and we can reason about certain norm violations that are more likely to boost the agent's expected utility.

(iii) From a macro perspective, we would like to further investigate the global benefits of norms (reduction of coordination problems, increments on the income from penalties) together with their enforcement cost (not considered in this research) so as to design and implement economically viable adaptive mechanisms for norm evolution in heterogeneous agent societies.

(iv) The last strand of future research will deal with cases where the agents explicitly account for the actions of other agents in the normative multiagent system. To this respect we intend to look into the field of stochastic games [105] and how they can be used to further generalize our approach.

Appendix A

Publications

The publications listed below are direct consequence of the evolution of this thesis:

- *Using Normative Markov Decision Processes for evaluating electronic contracts*, in collaboration with Sascha Ossowski, Michael Luck and Simon Miles, published in AI Communications (2012) [50].

- *Representing and evaluating electronic contracts with Normative Markov Decision Processes*, in collaboration with Sascha Ossowski, Michael Luck and Simon Miles, published in the 1st Conference on Agreement Technologies (2012) [49].

- *Normative Agents*, in collaboration with Michael Luck, Samhar Mahmoud, Felipe Meneguzzi, Martin Kollingbaum, Tim Norman and Natalia Criado, published in Agreement Technologies book (2012) [76].

- *Normative Reasoning with an Adaptive Self–interested Agent Model Based on Markov Decision Processes*, in collaboration with Holger Billhardt and Sascha Ossowski, published in the 12th Ibero-American Conference on AI (2010) [46].

- *Reasoning about Norm Compliance with Rational Agents*, in collaboration with Holger Billhardt and Sascha Ossowski, published in the 19th European Conference on Artificial Intelligence (2010) [47].

- *Behavior Adaptation in RMAS: An Agent Architecture based on MDPs*, in collaboration with Holger Billhardt and Sascha Ossowski, published in the 20th European Meeting on Cybernetics and Systems Research (2010) [45].

- *Designing Organized Multiagent Systems through MDPs*, in collaboration with Roberto Centeno, Holger Billhardt and Sascha Ossowski, published in the 7th German Conference on Multi-Agent System Technologies (2009) [48].

Appendix B

Motion problem's NMDP

This appendix specifies the NMDP input utilized the experiments in Chapter 5. For the sake of space, the capability function $\mathcal{C}$, the state-transition function $\mathcal{T}$ and the reward function $\mathcal{R}$ are partially shown.

NMDP = ⟨S,A,C,T,R,N,D⟩

```
//======================
// State space
//======================
```

f_1 = Street
f_2 = Cell
f_3 = Direction
f_4 = Status

V_{f_1} = {00,01, ...07,NW,NE,SW,SE,WN,WS,EN,ES}
V_{f_2} = {00,01, ...23,Gateway}
V_{f_3} = {Up,Right,Down,Left}
V_{f_4} = {Still,Moving}

$S = V_{f_1} \times V_{f_2} \times V_{f_3} \times V_{f_4}$

```
//======================
// Actions
// s ∈ {00,01, ...07}
// r ∈ {cw,ccw}
//======================
```

A = {Start(s),Move,Turn(r),Stop,End}

```
//======================
// Capabilities
//======================
C((Street=00,Cell=00,Direction=Up,Status=Still)) = {Move,Turn(r),End}
C((Street=00,Cell=00,Direction=Up,Status=Moving)) = {Move,Turn(r),Stop,End}
C((Street=00,Cell=00,Direction=Right,Status=Still)) = {Move,Turn(r),End}
C((Street=00,Cell=00,Direction=Right,Status=Moving)) = {Move,Turn(r),Stop,End}
C((Street=00,Cell=00,Direction=Down,Status=Still)) = {Move,Turn(r),End}
C((Street=00,Cell=00,Direction=Down,Status=Moving)) = {Move,Turn(r),Stop,End}
C((Street=00,Cell=00,Direction=Left,Status=Still)) = {Move,Turn(r),End}
C((Street=00,Cell=00,Direction=Left,Status=Moving)) = {Move,Turn(r),Stop,End}
...
C((Street=00,Cell=01,Direction=Up,Status=Still)) = {Move,Turn(r)}
C((Street=00,Cell=01,Direction=Up,Status=Moving)) = {Move,Turn(r),Stop}
C((Street=00,Cell=01,Direction=Right,Status=Still)) = {Move,Turn(r)}
C((Street=00,Cell=01,Direction=Right,Status=Moving)) = {Move,Turn(r),Stop}
C((Street=00,Cell=01,Direction=Down,Status=Still)) = {Move,Turn(r)}
C((Street=00,Cell=01,Direction=Down,Status=Moving)) = {Move,Turn(r),Stop}
C((Street=00,Cell=01,Direction=Left,Status=Still)) = {Move,Turn(r)}
C((Street=00,Cell=01,Direction=Left,Status=Moving)) = {Move,Turn(r),Stop}
...
C((Street=07,Cell=23,Direction=Up,Status=Still)) = {Move,Turn(r),End}
C((Street=07,Cell=23,Direction=Up,Status=Moving)) = {Move,Turn(r),Stop,End}
C((Street=07,Cell=23,Direction=Right,Status=Still)) = {Move,Turn(r),End}
C((Street=07,Cell=23,Direction=Right,Status=Moving)) = {Move,Turn(r),Stop,End}
C((Street=07,Cell=23,Direction=Down,Status=Still)) = {Move,Turn(r),End}
C((Street=07,Cell=23,Direction=Down,Status=Moving)) = {Move,Turn(r),Stop,End}
C((Street=07,Cell=23,Direction=Left,Status=Still)) = {Move,Turn(r),End}
C((Street=07,Cell=23,Direction=Left,Status=Moving)) = {Move,Turn(r),Stop,End}
...
C((Street=NW,Cell=Gateway,Direction=Up,Status=Still)) = {Start(s)}
C((Street=NW,Cell=Gateway,Direction=Up,Status=Moving)) = { }
C((Street=NW,Cell=Gateway,Direction=Right,Status=Still)) = {Start(s)}
C((Street=NW,Cell=Gateway,Direction=Right,Status=Moving)) = { }
C((Street=NW,Cell=Gateway,Direction=Down,Status=Still)) = {Start(s)}
C((Street=NW,Cell=Gateway,Direction=Down,Status=Moving)) = { }
C((Street=NW,Cell=Gateway,Direction=Left,Status=Still)) = {Start(s)}
C((Street=NW,Cell=Gateway,Direction=Left,Status=Moving)) = { }
...

//======================
// State-transitions
//======================
...
T((Street=00,Cell=00,Direction=Right,Status=Still),Move,
  (Street=00,Cell=00,Direction=Right,Status=Still)) = 0.01
T((Street=00,Cell=00,Direction=Right,Status=Still),Move,
  (Street=00,Cell=01,Direction=Right,Status=Moving)) = 0.99
```

```
T((Street=00,Cell=00,Direction=Right,Status=Still),Turn(cw),
  (Street=00,Cell=00,Direction=Right,Status=Still)) = 0.01
T((Street=00,Cell=00,Direction=Right,Status=Still),Turn(cw),
  (Street=00,Cell=00,Direction=Down,Status=Moving)) = 0.99
T((Street=00,Cell=00,Direction=Right,Status=Still),Turn(ccw),
  (Street=00,Cell=00,Direction=Right,Status=Still)) = 0.01
T((Street=00,Cell=00,Direction=Right,Status=Still),Turn(ccw),
  (Street=00,Cell=00,Direction=Up,Status=Moving)) = 0.99
...
T((Street=00,Cell=00,Direction=Left,Status=Moving),End,
  (Street=NW,Cell=Gateway,Direction=Left,Status=Moving)) = 1.00
...
T((Street=NW,Cell=Gateway,Direction=Right,Status=Still),Start(00),
  (Street=00,Cell=00,Direction=Right,Status=Moving)) = 1.00
T((Street=NW,Cell=Gateway,Direction=Right,Status=Still),Start(01),
  (Street=01,Cell=00,Direction=Right,Status=Moving)) = 1.00
...

//======================
// Rewards
//======================

...
R((Street=00,Cell=00,Direction=Right,Status=Still),Move,
  (Street=00,Cell=00,Direction=Right,Status=Still)) = -0.01R_N
R((Street=00,Cell=00,Direction=Right,Status=Still),Move,
  (Street=00,Cell=01,Direction=Right,Status=Moving)) = -0.01R_N
R((Street=00,Cell=00,Direction=Right,Status=Still),Turn(cw),
  (Street=00,Cell=00,Direction=Right,Status=Still)) = -0.01R_N
R((Street=00,Cell=00,Direction=Right,Status=Still),Turn(cw),
  (Street=00,Cell=00,Direction=Down,Status=Moving)) = -0.01R_N
R((Street=00,Cell=00,Direction=Right,Status=Still),Turn(ccw),
  (Street=00,Cell=00,Direction=Right,Status=Still)) = -0.01R_N
R((Street=00,Cell=00,Direction=Right,Status=Still),Turn(ccw),
  (Street=00,Cell=00,Direction=Up,Status=Moving)) = -0.01R_N
...
R((Street=00,Cell=00,Direction=Left,Status=Moving),End,
  (Street=NW,Cell=Gateway,Direction=Left,Status=Moving)) = 0.6R_N
...
R((Street=NW,Cell=Gateway,Direction=Right,Status=Still),Start(00),
  (Street=00,Cell=00,Direction=Right,Status=Moving)) = -0.01R_N
R((Street=NW,Cell=Gateway,Direction=Right,Status=Still),Start(01),
  (Street=01,Cell=00,Direction=Right,Status=Moving)) = -0.01R_N
...
```

```
//======================
// Norms
//======================
```

$N = \{q_{01}, q_{02}, q_{03}, \ldots q_{24}\}$

q_{01} = ⟨01, Obligation, ⊤,
(Street=00),
(Direction=Right),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Direction=Right)} } ⟩⟩

q_{02} = ⟨02, Obligation, ⊤,
(Street=01),
(Direction=Left),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Direction=Left)} } ⟩⟩

q_{03} = ⟨03, Obligation, ⊤,
(Street=02),
(Direction=Right),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Direction=Right)} } ⟩⟩

q_{04} = ⟨04, Obligation, ⊤,
(Street=03),
(Direction=Left),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Direction=Left)} } ⟩⟩

q_{05} = ⟨05, Obligation, ⊤,
(Street=04),
(Direction=Up),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Direction=Up)} } ⟩⟩

q_{06} = ⟨06, Obligation, ⊤,
(Street=05),
(Direction=Down),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Direction=Down)} } ⟩⟩

q_{07} = ⟨07, Obligation, ⊤,
(Street=06),
(Direction=Up),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Direction=Up)} } ⟩⟩

q_{08} = ⟨08, Obligation, ⊤,
(Street=07),
(Direction=Down),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Direction=Down)} } ⟩⟩

q_{09} = ⟨09, Prohibition, ⊤,
(Street=00) ∧ (Direction=Right) ∧ (Cell=03),
(Status=Moving),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Status=Still)} } ⟩⟩

q_{10} = ⟨10, Prohibition, ⊤,
(Street=00) ∧ (Direction=Right) ∧ (Cell=08),
(Status=Moving),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Status=Still)} } ⟩⟩

q_{11} = ⟨11, Prohibition, ⊤,
(Street=00) ∧ (Direction=Right) ∧ (Cell=13),
(Status=Moving),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Status=Still)} } ⟩⟩

q_{12} = ⟨12, Prohibition, ⊤,
(Street=00) ∧ (Direction=Right) ∧ (Cell=18),
(Status=Moving),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Status=Still)} } ⟩⟩

q_{13} = ⟨13, Prohibition, ⊤,
(Street=01) ∧ (Direction=Left) ∧ (Cell=05),
(Status=Moving),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Status=Still)} } ⟩⟩

q_{14} = ⟨14, Prohibition, ⊤,
(Street=01) ∧ (Direction=Left) ∧ (Cell=10),
(Status=Moving),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Status=Still)} } ⟩⟩

q_{15} = ⟨15, Prohibition, ⊤,
(Street=01) ∧ (Direction=Left) ∧ (Cell=15),
(Status=Moving),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Status=Still)} } ⟩⟩

q_{16} = ⟨16, Prohibition, ⊤,
(Street=01) ∧ (Direction=Left) ∧ (Cell=20),
(Status=Moving),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Status=Still)} } ⟩⟩

q_{17} = ⟨17, Prohibition, ⊤,
(Street=02) ∧ (Direction=Right) ∧ (Cell=03),
(Status=Moving),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Status=Still)} } ⟩⟩

q_{18} = ⟨18, Prohibition, ⊤,
(Street=02) ∧ (Direction=Right) ∧ (Cell=08),
(Status=Moving),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Status=Still)} } ⟩⟩

q_{19} = ⟨19, Prohibition, ⊤,
(Street=02) ∧ (Direction=Right) ∧ (Cell=13),
(Status=Moving),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Status=Still)} } ⟩⟩

q_{20} = ⟨20, Prohibition, ⊤,
(Street=02) ∧ (Direction=Right) ∧ (Cell=18),
(Status=Moving),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Status=Still)} } ⟩⟩

q_{21} = ⟨21, Prohibition, ⊤,
(Street=03) ∧ (Direction=Left) ∧ (Cell=05),
(Status=Moving),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Status=Still)} } ⟩⟩

q_{22} = ⟨22, Prohibition, ⊤,
(Street=03) ∧ (Direction=Left) ∧ (Cell=10),
(Status=Moving),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Status=Still)} } ⟩⟩

q_{23} = ⟨23, Prohibition, ⊤,
(Street=03) ∧ (Direction=Left) ∧ (Cell=15),
(Status=Moving),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Status=Still)} } ⟩⟩

q_{24} = ⟨24, Prohibition, ⊤,
(Street=03) ∧ (Direction=Left) ∧ (Cell=20),
(Status=Moving),
⟨ { ⊤ → −λR_N },
{ ⊤ → {(Status=Still)} } ⟩⟩

```
//======================
// Detection function
//======================
```

...

D(q_{01},(Street=00,Cell=00,Direction=Up,Status=Still)) = β
D(q_{01},(Street=00,Cell=00,Direction=Up,Status=Moving)) = β
D(q_{01},(Street=00,Cell=00,Direction=Down,Status=Still)) = β
D(q_{01},(Street=00,Cell=00,Direction=Down,Status=Moving)) = β

...

Appendix C

Time consumption results

This appendix presents detailed reports about the time consumption during the reasoning processes of the agent models, from which the time consumption data shown in Section 5.5.4 have been obtained[1]. These reports have been generated with the Java Interactive Profiler (JIP)[2], a profiling information tool.

The columns of a JIP output file are, from left to right:

- *Count*: The number of times that the given method is called by the enclosing method.
- *Total time*: The time, in milliseconds, that was taken up by the execution of the given method.
- *Net Time*: The amount of time that was actually spent performing the given method if you factor out the total time taken by calling other (listed) methods. It important to note that the net time is the total time less the sum of the total times for all of the listed child methods.
- *Location*: The method called.

[1]The experiments have been performed in a MacBook Pro, 2.53GHz Intel Core 2 Duo processor, 4GB DDR3, 250GB 5400-rpm hard drive, OS X Snow Leopard, Java Platform SE 6.

[2]The Java Interactive Profiler is a high performance, low overhead profiler that is written entirely in Java and distributed under the BSD license (freeware). More information about this tool can be found at http://jiprof.sourceforge.net.

This appendix contains six reports, which are explained as follows:

Report (i): As defined in Chapter 4, the *normative reasoning* of the *self-interested* agent model is performed by means of Algorithm 3 and Algorithm 4, which identify violating states and represent sanctions, respectively. This report highlights in red the total time consumed by these algorithms in the experiments carried out in Chapter 5.

Report (ii): The *normative reasoning* of the *norm-compliant* agent model, as defined in Chapter 4, is executed by means of Algorithm 5 and Algorithm 6. This report highlights in red the total time consumed by these algorithms in the experiments made in Chapter 5.

Report (iii): This report shows the time consumed by the *Value Iteration* algorithm used in the policy construction of the *self-interested* agents in the experiments made in Chapter 5. This report highlights in red the *total time* spent by the algorithm and the number of *value iterations* counted during its execution.

Report (iv): This report shows the time consumed by the *Value Iteration* algorithm used in the policy construction of the *norm-compliant* agents in the experiments made in Chapter 5. This report highlights in red the *total time* spent by the algorithm and the number of *value iterations* counted during its execution.

Report (v): This report shows the time consumed by the *Modified Policy Iteration* algorithm used in the policy construction of the *self-interested* agents in the experiments made in Chapter 5. This report highlights in red the *total time* spent by the algorithm and the number of *policy iterations* counted during its execution.

Report (vi): This report shows the time consumed by the *Modified Policy Iteration* algorithm used in the policy construction of the *norm-compliant* agents in the experiments made in Chapter 5. This report highlights in red the *total time* spent by the algorithm and the number of *policy iterations* counted during its execution.

Report (i) - Normative reasoning of the self-interested agent model.

Count	Time Total	Time Net	Location
1	2856,3	201,4	+--Algorithm3:run
76800	11,6	11,6	| +--Norm:getNormID
76800	122,3	109,0	| +--NMDP:getDetection
76800	13,3	13,3	| | +--Detection:getProbability
78432	2520,9	1651,5	| +--Algorithm3:match
392160	81,4	81,4	| | +--MDP:getFeatureLength
313728	50,2	50,2	| | +--MDP:getFeature
313728	49,9	49,9	| | +--Feature:getName
313728	138,9	138,9	| | +--MDP:getState
313728	51,7	51,7	| | +--State:getFeatureValue
3288	311,1	311,1	| | +--Algorithm3:parseExpression
78432	186,2	186,2	| | +--Algorithm3:evaluateExpression
1	1441,4	473,8	+--Algorithm4:run
80001	27,8	27,8	| +--NMDP:getNormLength
1	352,5	352,5	| +--NMDP:deepcopy
16085	2,5	2,5	| +--MDP:getStateLength
1216	26,9	1,3	| +--NMDP:getDetection
1216	25,6	25,6	| | +--Detection:getProbability
1216	61,4	61,4	| +--State:isAbsorbing
11040	3,9	3,9	| +--MDP:getActionLength
9728	2,2	2,2	| +--MDP:isCapable
38640	71,8	59,6	| +--MDP:getOutcomeLength
38640	6,1	6,1	| | +--MDP:getState
38640	6,1	6,1	| | +--State:isAbsorbing
9248	2,0	2,0	| +--MDP:getOutcomeArray
8704	11,7	10,0	| +--MDP:getProbability
8704	1,7	1,7	| | +--Outcome:getProbability
17408	21,2	17,1	| +--MDP:setProbability
17408	4,1	4,1	| | +--Outcome:setProbability
29232	37,8	32,6	| +--MDP:getOutcome
29232	5,2	5,2	| | +--Outcome:getState
8704	11,2	8,2	| +--MDP:getReward
8704	3,1	3,1	| | +--Outcome:getReward
4816	31,5	23,4	| +--State:<init>
4816	2,1	2,1	| | +--State:setIndex
4816	3,3	3,3	| | +--State:isAbsorbing
4816	149,4	137,4	| +--Algorithm4:updateState
14512	2,4	2,4	| | +--MDP:getFeatureLength
14512	2,3	2,3	| | +--MDP:getFeature
14512	2,2	2,2	| | +--Feature:getName
9632	4,2	4,2	| | +--Algorithm4:updateState
4816	1,0	1,0	| | +--State:updateFeatureValue
4816	111,5	60,3	| +--MDP:getStateIndex
19264	22,9	22,9	| | +--State:getFeatureValue
19264	20,6	20,6	| | +--MDP:getFeature
19264	4,7	4,7	| | +--Feature:getNumberValues
19264	3,1	3,1	| | +--MDP:getFeatureLength
8704	9,4	8,0	| +--MDP:setReward
8704	1,4	1,4	| | +--Outcome:setReward
4480	28,8	12,5	| +--Outcome:<init>
4480	15,0	15,0	| | +--Outcome:setReward
4480	1,2	1,2	| +--MDP:setOutcomeArray

Report (ii) - Normative reasoning of the norm-compliant agent model.

Time			
Count	Total	Net	Location
1	2746,4	75,4	+--Algorithm5:run
78432	2670,9	1746,3	| +--Algorithm5:match
392160	62,0	62,0	| | +--MDP:getFeatureLength
313728	81,1	81,1	| | +--MDP:getFeature
313728	90,7	90,7	| | +--Feature:getName
313728	60,5	60,5	| | +--MDP:getState
313728	70,8	70,8	| | +--State:getFeatureValue
3288	282,1	282,1	| | +--Algorithm5:parseExpression
78432	277,4	277,4	| | +--Algorithm5:evaluateExpression
1	414,0	11,0	+--Algorithm6:run
1	400,2	400,2	| +--MDP:deepcopy

Report (iii) - Policy construction of the self-interested agent model using the Value Iteration algorithm

Time			
Count	Total	Net	Location
1	12137,3	953,9	+--ValueIteration:solve
140890	26,0	26,0	| +--MDP:getStateLength
1	18,9	18,0	| +--UtilityFunction:<init>
4817	23,6	17,3	| +--MDP:isCapable
16840	6,4	6,4	| | +--MDP:isCapable
207100	36,5	36,5	| +--MDP:getState
42	1,7	1,7	| +--History:startIteration
42	317,0	317,0	| +--UtilityFunction:deepcopy
137643	28,8	28,8	| +--State:isAbsorbing
67872	10,2	10,2	| +--State:isCompliant
203616	41,5	41,5	| +--UtilityFunction:getUtility
69488	10620,7	4879,1	| +--ValueIteration:calculateMeuAction
625392	115,3	115,3	| | +--MDP:getActionLength
555904	104,0	104,0	| | +--MDP:isCapable
1243904	2626,2	2142,6	| | +--MDP:getOutcomeLength
1243904	260,3	260,3	| | | +--MDP:getState
1243904	223,3	223,3	| | | +--State:isAbsorbing
694880	880,4	717,5	| | +--MDP:getProbability
694880	163,0	163,0	| | | +--Outcome:getProbability
694880	860,8	736,9	| | +--MDP:getOutcome
694880	123,8	123,8	| | | +--Outcome:getState
694880	153,2	153,2	| | +--UtilityFunction:getUtility
694880	148,4	148,4	| | +--Utility:getUtility
694880	835,0	712,8	| | +--MDP:getReward
694880	122,2	122,2	| | | +--Outcome:getReward
69488	18,3	18,3	| | +--ValueIteration$MeuAction:<init>
67872	11,2	11,2	| +--ValueIteration$MeuAction:getEU
67872	12,8	12,8	| +--Utility:setUtility
135744	30,2	30,2	| +--Utility:getUtility
42	1,1	1,0	| +--History:stopIteration
3201	1,1	1,1	| +--Decision:<init>

Report (iv) - Policy construction of the norm-compliant agent model using the Value Iteration algorithm

Time			
Count	Total	Net	Location
1	2603,6	604,6	+--ValueIteration:solve
140890	25,3	25,3	| +--MDP:getStateLength
1	7,5	6,8	| +--UtilityFunction:<init>
4753	22,0	17,4	| +--MDP:isCapable
17784	4,6	4,6	| | +--MDP:isCapable
204476	62,4	62,4	| +--MDP:getState
42	2,5	2,5	| +--History:startIteration
42	330,2	330,2	| +--UtilityFunction:deepcopy
137643	32,4	32,4	| +--State:isAbsorbing
65184	13,2	13,2	| +--State:isCompliant
48384	9,7	9,7	| +--UtilityFunction:getUtility
17680	1468,5	706,1	| +--ValueIteration:calculateMeuAction
159120	49,0	49,0	| | +--MDP:getActionLength
141440	28,7	28,7	| | +--MDP:isCapable
155792	346,8	284,0	| | +--MDP:getOutcomeLength
155792	28,2	28,2	| | | +--MDP:getState
155792	34,6	34,6	| | | +--State:isAbsorbing
76080	96,6	82,6	| | +--MDP:getProbability
76080	14,0	14,0	| | | +--Outcome:getProbability
76080	108,2	86,2	| | +--MDP:getOutcome
76080	22,0	22,0	| | | +--Outcome:getState
76080	15,5	15,5	| | +--UtilityFunction:getUtility
76080	28,9	28,9	| | +--Utility:getUtility
76080	84,5	71,6	| | +--MDP:getReward
76080	12,9	12,9	| | | +--Outcome:getReward
17680	4,3	4,3	| | +--ValueIteration$MeuAction:<init>
16128	2,7	2,7	| +--ValueIteration$MeuAction:getEU
16128	8,5	8,5	| +--Utility:setUtility
32256	9,4	9,4	| +--Utility:getUtility
42	1,8	1,7	| +--History:stopIteration
3201	1,1	1,1	| +--Decision:<init>

Report (v) - Policy construction of the self-interested agent model using the Modified Policy Iteration algorithm

Count	Time Total	Time Net	Location
1	3836,0	142,1	+--PolicyIteration:solve
25617	3,9	3,9	| +--MDP:getStateLength
48015	9,0	9,0	| +--MDP:getState
25608	3,9	3,9	| +--State:isAbsorbing
1616	4,5	2,4	| +--MDP:isCapable
2080	2,1	2,1	| | +--MDP:isCapable
1	28,0	20,7	| +--Policy:<init>
3776	2,6	2,2	| | +--MDP:isCapable
3201	2,3	2,3	| | +--Policy:setDecision
1	3,9	3,3	| +--UtilityFunction:<init>
7	1,5	1,5	| +--History:startIteration
7	2196,0	1143,2	| +--EvaluationVI:evaluate
112070	17,4	17,4	| | +--MDP:getStateLength
168595	27,7	27,7	| | +--MDP:getState
112035	17,1	17,1	| | +--State:isAbsorbing
56560	8,4	8,4	| | +--State:isCompliant
56560	8,3	8,3	| | +--Policy:getDecision
56560	10,7	10,7	| | +--Decision:getAction
255410	452,5	373,9	| | +--MDP:getOutcomeLength
255410	41,8	41,8	| | | +--MDP:getState
255410	36,9	36,9	| | | +--State:isAbsorbing
142290	155,5	127,9	| | +--MDP:getProbability
142290	27,6	27,6	| | | +--Outcome:getProbability
142290	142,4	119,6	| | +--MDP:getOutcome
142290	22,7	22,7	| | | +--Outcome:getState
198850	32,8	32,8	| | +--UtilityFunction:getUtility
142290	28,8	28,8	| | +--Utility:getUtility
142290	141,8	121,3	| | +--MDP:getReward
142290	20,5	20,5	| | | +--Outcome:getReward
56560	9,3	9,3	| | +--Utility:setUtility
22407	3,5	3,5	| +--State:isCompliant
11312	1430,7	656,4	| +--PolicyIteration:calculateMeuAction
101808	15,5	15,5	| | +--MDP:getActionLength
90496	15,9	15,9	| | +--MDP:isCapable
202496	355,9	294,0	| | +--MDP:getOutcomeLength
202496	32,6	32,6	| | | +--MDP:getState
202496	29,3	29,3	| | | +--State:isAbsorbing
113120	120,0	101,6	| | +--MDP:getProbability
113120	18,4	18,4	| | | +--Outcome:getProbability
113120	113,5	95,5	| | +--MDP:getOutcome
113120	18,0	18,0	| | | +--Outcome:getState
113120	18,5	18,5	| | +--UtilityFunction:getUtility
113120	18,1	18,1	| | +--Utility:getUtility
113120	115,1	96,3	| | +--MDP:getReward
113120	18,7	18,7	| | | +--Outcome:getReward
11312	1,9	1,9	| | +--PolicyIteration$MeuAction:<init>
11312	2,0	2,0	| +--PolicyIteration$MeuAction:getEU
13944	2,2	2,2	| +--UtilityFunction:getUtility
11319	1,9	1,9	| +--Utility:getUtility

Report (vi) - Policy construction of the norm-compliant agent model using the Modified Policy Iteration algorithm

Time			
Count	Total	Net	Location
1	1601,9	118,3	+--PolicyIteration:solve
32021	6,3	6,3	| +--MDP:getStateLength
49891	9,8	9,8	| +--MDP:getState
21066	10,9	10,9	| +--State:isAbsorbing
1568	2,7	2,3	| +--MDP:isCapable
1	37,9	23,0	| +--Policy:<init>
3203	5,3	5,3	| | +--MDP:getStateLength
3201	1,8	1,8	| | +--MDP:getState
3201	1,8	1,8	| | +--State:isAbsorbing
3320	4,3	3,7	| | +--MDP:isCapable
1	8,4	4,8	| +--UtilityFunction:<init>
3201	3,6	3,6	| | +--Utility:<init>
9	1,9	1,9	| +--History:startIteration
9	1120,8	691,6	| +--EvaluationVI:evaluate
144090	26,7	26,7	| | +--MDP:getStateLength
213885	36,6	36,6	| | +--MDP:getState
144045	47,6	47,6	| | +--State:isAbsorbing
69840	11,6	11,6	| | +--State:isCompliant
17280	3,7	3,7	| | +--Policy:getDecision
17280	3,9	3,9	| | +--Decision:getAction
67460	139,0	106,1	| | +--MDP:getOutcomeLength
67460	22,4	22,4	| | | +--MDP:getState
67460	10,4	10,4	| | | +--State:isAbsorbing
32900	39,3	29,1	| | +--MDP:getProbability
32900	10,2	10,2	| | | +--Outcome:getProbability
32900	50,8	31,8	| | +--MDP:getOutcome
32900	19,0	19,0	| | | +--Outcome:getState
50180	15,1	15,1	| | +--UtilityFunction:getUtility
32900	12,0	12,0	| | +--Utility:getUtility
32900	40,0	35,0	| | +--MDP:getReward
32900	5,0	5,0	| | | +--Outcome:getReward
17280	2,8	2,8	| | +--Utility:setUtility
28809	6,9	6,9	| +--State:isCompliant
3456	267,9	125,9	| +--PolicyIteration:calculateMeuAction
31104	21,2	21,2	| | +--MDP:getActionLength
27648	5,1	5,1	| | +--MDP:isCapable
28800	55,7	43,5	| | +--MDP:getOutcomeLength
28800	4,7	4,7	| | | +--MDP:getState
28800	7,6	7,6	| | | +--State:isAbsorbing
14112	16,0	13,4	| | +--MDP:getProbability
14112	2,7	2,7	| | | +--Outcome:getProbability
14112	16,3	12,7	| | +--MDP:getOutcome
14112	3,7	3,7	| | | +--Outcome:getState
14112	2,9	2,9	| | +--UtilityFunction:getUtility
14112	8,9	8,9	| | +--Utility:getUtility
14112	15,0	12,6	| | +--MDP:getReward
14112	2,4	2,4	| | | +--Outcome:getReward
3463	2,4	2,4	| +--Utility:getUtility

Appendix D

Resumen en castellano

Todas las teorías son legítimas y ninguna tiene importancia. Lo que importa es lo que se hace con ellas.

Jorge Luis Borges

D.1 Antecedentes

Diariamente nos enfrentamos a problemas en los cuales debemos hacer decisiones secuenciales. Por ejemplo, cuando conducimos nuestro coche desde una dirección hasta otra, tenemos que decidir la ruta; cuando una empresa tiene la intención de lanzar un nuevo producto en el mercado, tiene que decidir de que proveedor comprar los componentes necesario para construir el nuevo producto. Cuando un estudiante empieza un doctorado, tiene que decidir su linea de investigación.

La esencia de las decisiones secuenciales es que nuestras elecciones hoy dependen de las decisiones que hicimos en el pasado, y el resultado de tales decisiones tomadas ahora pueden afectar las decisiones que haremos en el futuro. Esto quiere decir que nuestra mejor decisión depende de las situaciones futuras y de como las afrontamos. En las ciencias de la computación, el problema de desarrollar mecanismos para que agentes puedan tomar decisiones secuenciales ha sido extensamente investigado por medio de *Markov Decision Processes* (MDPs) [6, 61, 95, 10, 93], los cuales han demostrado tener gran adaptabilidad para análisis cuantitativo del rendimiento de los agentes. En principio, MDPs pueden ser usados para generar y evaluar decisiones secuenciales, aunque el problema de computar un plan óptimo es muy complejo [90, 75].

En algunas situaciones, normas afectarán significativamente la toma decisiones secuenciales. Ellas influencian nuestras decisiones a través de reglas que restringen nuestras opciones con el objetivo de fomentar algún tipo de coordinación. Por ejemplo, cuando conducimos nuestro coche por las calles, existen normas regulando el tráfico con el objetivo de evitar accidentes; cuando las empresas compran productos de otras empresas, las interacciones entre ellas casi siempre son reguladas por contratos previamente establecidos; cuando un estudiante accede a un programa de doctorado, se esperan que él cumpla con las normas de la universidad.

En estos ejemplos, se supone que los participantes son autónomos para decidir si cumplen o no las normas. Tomar tales decisiones en entornos sencillos no suele ser un problema. Aun así, la situación cambia en entornos complejos donde hay incertidumbre. En este caso, los agentes no siempre estarán seguros de los resultados, recompensas y costes de sus acciones, bien como no siempre estarán seguros de cuando las violaciones de normas serán detectadas y castigadas.

En las últimas décadas, varios modelos de agentes para permitir razonamientos normativos fueron propuestos [28, 42, 24, 68, 2, 123]. Una de las principales ventajas de estos modelos de agente normativo vienen del hecho que podemos usarlos para desarrollar sistemas donde las interacciones entre los agentes pueden ser coordinadas por medio de normas que restringen sus actividades [107, 3].

Estudiando el estado del arte en MDPs y modelos de agente normativo hemos observado la inexistencia de trabajos combinando ambos abordajes para la toma de decisión. Esta tesis se enfoca en la clase de problemas en los cuales agentes utilitarios modelados como MDPs toman decisiones secuenciales en entornos regulados por normas. Creemos que con esto, cada una de estas áreas de investigación pueden beneficiarse de los puntos positivos de la otra. Por un lado, las normas nos podrían ayudar a moldear el comportamiento de los agentes, fomentando la coordinación de actividades y reduciendo el coste de computar planes de acción. Por otro lado, los MDPs pueden ayudar con su capacidad para realizar evaluaciones cuantitativas, permitiendo de este modo evaluar el impacto de las normas en la utilidad de los agentes.

D.2 Objetivos

El objetivo principal de esta tesis es desarrollar y evaluar métodos computacionales de agentes racionales capaces de generar planes de acción complejos en entornos dinámicos y estocásticos regulados por normas.

El objetivo principal puede ser divido en los siguientes objetivos específicos:

(i) Formalizar un *framework* que permita la especificación de agentes utilitarios normativos en entornos estocásticos donde hay normas regulando sus actividades.

(ii) Desarrollar modelos de razonamiento normativo en entornos estocásticos para permitir que agentes utilitarios normativos puedan incorporar las normas en su base de conocimiento y decidir cumplir o no las normas.

(iii) Investigar el impacto de los modelos de razonamiento normativo propuestos en las decisiones secuenciales y utilidades de los agentes.

(iv) Desde una perspectiva global, estudiar como las normas pueden ser usadas para coordinar sociedades de agentes utilitarios.

D.3 Metodología

Para cumplir el objetivo (i), hemos extendido los *Markov Decision Processes*, incluyendo estructuras normativas y una distribución de probabilidad que determina la probabilidad de que las violaciones de normas sean detectadas. La formalización de nuestro *framework* (NMDPs) ha sido hecha por medio de la teoría de conjuntos, y nuestra noción de normas tiene su raíz en trabajos previos en sistemas normativos informáticos.

Para desarrollar los métodos de razonamiento normativo propuesto en el objetivo (ii), hemos utilizado el *framework* del objetivo (i) como base de conocimiento. La estructura del *framework* nos permitió formular algoritmos generales y eficientes para implementar diferentes nociones de racionalidad. Una vez codificados los algoritmos, les hemos integrado en dos modelos de agentes, llamados *self-interested* y *norm-compliant*.

Los objetivos (iii) y (iv) fueron cumplidos a través de experimentaciones basadas en simulaciones realizadas en entornos multiagente normativos.

D.4 Conclusiones

D.4.1 Contribuciones

Las principales contribuciones de la presente tesis son las siguientes:

(i) *Generalización de los MDPs para permitir el razonamiento sobre normas.*
En el Capítulo 3, nos hemos centrado en la representación de conocimiento en agentes racionales capaces de percibir normas. Con este fin, hemos introducido los NMDPs, que permiten la representación de estructuras normativas y distribuciones de probabilidad con respecto a la detección de violaciones de normas. Los NMDPs proporcionan un método para constatar el cumplimiento de las normas en los diferentes estados, y verificar la consistencia de normas y sanciones (detección de conflictos). Además, hemos demostrado como representaciones factorizadas de un espacio de estados pueden ser explotadas para que se pueda codificar normas y modelos de detección de un modo compacto.

En un contexto mas amplio, creemos que los NMDPs pueden ser utilizados como base para el desarrollo de nuevos modelos de razonamiento normativos, en particular, permitiendo la especificación de algoritmos generales para razonamiento normativo en entornos estocásticos.

(ii) *Modelos de razonamiento normativo.*
Para abordar el problema de como razonar sobre normas con los NMDPs introducidos en el Capítulo 3, hemos propuesto dos modelos de razonamiento normativo en el Capítulo 4. En particular, esos modelos son especificados por medio de algoritmos e incorporados en dos modelos de agentes. Los agentes egoístas incorporan las normas en su funciones de transición y recompensa, y de ese modo son capaces de razonar sobre posibles violaciones de normas. Por otro lado, los agentes que cumplen las normas utilizan las estructuras normativas para eliminar aquellos estados que violan las normas, generando así MDP reducidos.

En esos modelos de agentes, hemos cubierto todas las propiedades utilizadas en el análisis hecho en la Sección 2.5: autonomía con respecto a normas, representaciones explicitas de normas y sanciones, modelo cuantitativo de decisión y planeamiento probabilístico. Al tomar ventaja de la sinergia entre estas propiedades, hemos posibilitado el desarrollo de agentes normativos capaces

de tomar decisiones secuenciales con el objetivo de maximizar sus utilidades en entornos estocásticos regulados por normas explícitamente representadas.

(iii) *Sinergia entre MDPs y normas.*
En el ámbito de la presente tesis, exploramos la combinación de los puntos fuertes nativos de los MDPs y de los sistemas normativos con el propósito de cubrir sus respectivas limitaciones individuales. Por un lado, usamos las ventajas analíticas de los MDPs para modelar el dominio de interacción y el proceso de toma de decisión del agente llevando en cuenta la incertidumbre en relación al entorno. Por otro lado, usamos las normas para fomentar la coordinación en las sociedades de agentes y acotar el espacio de búsqueda en la construcción de los planes con la intención de reducir el tiempo necesario para computar un plan óptimo.

(a) *Evaluación cuantitativa de normas.*
Desde una perspectiva local, nuestro modelos permiten que los agentes calculen el impacto de las normas en sus utilidades esperadas, teniendo en cuenta incertidumbre, recompensas y sanciones. Tal evaluación cuantitativa es particularmente vital en dominios donde el rendimiento de los agentes esta enlazado con métricas importantes tales como cantidad de accidentes en un entorno de movilidad o ganancias provenientes de actividades reguladas por contratos. Desde una perspectiva global, las evaluaciones cuantitativas fueron realizadas con un simulador de movilidad que mantiene registro de accidentes y retribuciones de multas relacionadas con violaciones de normas. En la Sección 5.5 enseñamos varios ejemplos de evaluaciones cuantitativas, donde se puede observar el impacto de varias normas en el comportamiento y utilidad de los agentes.

(b) *Reducciones en el tiempo necesario para computar un plan óptimo.*
Por medio del agente *norm-compliant*, hemos desarrollado un abordaje original para reducir el tiempo necesario para computar un plan óptimo. Nuestra técnica restringe el espacio de estados a una partición que respecta las normas. En la Sección 5.5.4 se presentan los resultados en los cuales podemos ver significantes reducciones en el tiempo necesario para generar un plan.

(c) *Coordinación de agentes basados en MDPs.*

Un MDP es un modelo de agente individual que no tiene conocimiento de otros agentes que se encuentran en su mismo entorno, tal como presentamos en el simulador de movilidad en la Sección 5.3. En ese contexto, las actividades de un agente pueden interferir con las actividades de los demás, y eso puede traer indeseables consecuencias. Suponiendo que el total cumplimiento de las normas da por hecho la inexistencia de problemas de coordinación, es posible coordinar sistemas multiagentes compuestos por agentes *self-interested* y/o *norm-compliant*. La diferencia es que con agentes *norm-compliant*, la coordinación es incondicionalmente soportada, y con agentes *self-interested*, la coordinación debe ser implementada por medio de sanciones adecuadas.

Perciba que la técnica de coordinación propuesta en esta tesis es flexible en el sentido que modificaciones en las normas no requieren un nueva implementación de los agentes. Una vez que las normas son reflejadas en los NMDPs, el nuevo plan puede ser generado automáticamente.

(iv) *Aplicaciones.*

Para validar nuestro abordaje, dos aplicaciones fueron desarrolladas en el ámbito de la presente tesis. En el Capítulo 5, hemos medido el rendimiento de los diferentes modelos de agente en relación a determinados parámetros controlados en un entorno de movilidad multiagente. Esos experimentos ilustran el tipo de evaluación cuantitativa que ideamos en sistemas multiagente normativos. El Capitulo 6 demuestra a través de un caso de estudio en un dominio de mercados secundarios aeroespaciales la aplicabilidad de nuestro abordaje para modelar y razonar sobre contratos.

D.4.2 Limitaciones

La fortaleza de esta tesis se basa en la sinergia entre áreas de investigación, que individualmente, presentan sus proprias limitaciones. Algunas de esas limitaciones fueron superadas en esta tesis, todavía otras siguen aguardando una solución satisfactoria, en gran parte debido a las restricciones impuestas en esta investigación de doctorado.

Los siguientes elementos enumeran las principales limitaciones de la presente tesis:

(i) Algunas limitaciones fueron heredadas de los MDPS, como por ejemplo, la imposibilidad de tratar observaciones parciales y acciones colectivas.

(ii) En esta tesis, se supone que las normas codificadas en los NMDPs son consistentes, o sea, no hay estados que simultáneamente cumplen con una norma y violan otra. Vasconselos et al. [116] estudian ese tema y proponen mecanismos para la resolución de conflictos normativos.

(iii) En determinadas aplicaciones, como por ejemplo en los contratos electrónicos, sería interesante tener un tratamiento mas sofisticado del tiempo (por ejemplo, *deadlines*). En esto trabajo, los *deadlines* son representados como propiedades de los estados que cambian de valor cuando el deadline es alcanzado. El trabajo de Broersen et al. [25, 11] en razonamiento normativo con *deadlines* puede ser usado como un punto de partida para una investigación mas detallada con los NMDPs.

D.4.3 Trabajos Futuros

Esta tesis ha abierto varias lineas para futuras investigaciones:

(i) Para abordar con problemas en los cuales los agentes no pueden observar directamente el estado actual, nos gustaría proponer los *Partially Observable Normative Markov Decision Processes* (PONMDPs), una extensión de los NMDPs incluyendo un conjunto de posibles observaciones y probabilidades condicionales para las observaciones. Además, tenemos la intención de representar transiciones y recompensas usando representaciones factorizadas (en esta tesis, el uso de factores esta restricta a los espacios de estados y modelos de detección) y remodelar los algoritmos de razonamiento propuestos en el Capitulo 4 para que puedan manipular esas representaciones compactas.

(ii) En el Capitulo 5 hemos visto que en determinadas situaciones los agentes *norm-compliant* superan el rendimiento de los agentes *self-interested*, y que en otras situaciones acontece al revés. Teniendo en cuenta esta observación, nos gustaría desarrollar un nuevo modelo de agentes capaz de considerar apenas un subconjunto específico de normas para violación. Con eso, esperamos ahorrar recursos computacionales (aún estaríamos explorando apenas parte del espacio de estados) y permitir el razonamiento sobre determinadas violaciones de normas que muy probablemente aumentan las utilidades de los agentes.

(iii) Desde una perspectiva global, queremos investigar mas a fondo los beneficios de las normas para la sociedad de agentes (reducciones en los problemas de coordinación, aumento de recaudaciones provenientes de sanciones contra violaciones de normas) junto con el coste económico de implementar dichos mecanismos normativos (no considerado en esta tesis). El propósito de tal investigación es propiciar el desarrollo de mecanismos económicamente viables y adaptables para la evolución de normas en sociedades de agentes heterogéneos.

(iv) La última linea de investigaciones futuras aborda casos donde los agentes tienen conocimiento de las posibles acciones de los demás agentes compartiendo su mismo entorno. Con respecto a esa linea, nos gustaría estudiar trabajos en juegos estocásticos [105] y como estos pueden ser usados para generalizar nuestro abordaje.

References

[1] Thomas Ågotnes, Wiebe van der Hoek, and Michael Wooldridge. Normative system games. In Edmund H. Durfee, Makoto Yokoo, Michael N. Huhns, and Onn Shehory, editors, *AAMAS*, pages 881–888. IFAAMAS, 2007.

[2] Giulia Andrighetto, Marco Campenni, Rosaria Conte, and Mario Paolucci. On the immergence of norms: a normative agent architecture. In *Proceedings of AAAI Symposium, Social and Organizational Aspects of Intelligence, Washington DC. Papers from the AAAI Fall Symposium, The AAAI Press, Menlo Park, California, Technical Report FS-07-04*, 2007.

[3] Mihai Barbuceanu. Role of obligations in multiagent coordination. *Applied Artificial Intelligence*, 13(1–2):11–38, 1999.

[4] Nicole Bauerle and Ulrich Rieder. *Markov Decision Processes with Applications to Finance*. Universitext. Springer, 2011.

[5] Raphen Becker, Shlomo Zilberstein, Victor R. Lesser, and Claudia V. Goldman. Transition-independent decentralized markov decision processes. In Rosenschein et al. [101], pages 41–48.

[6] Richard Bellman. A Markovian Decision Process. *Journal of Mathematics and Mechanics*, 6:679–684, 1957.

[7] Curt A. Bererton, Geoffrey J. Gordon, and Sebastian Thrun. Auction mechanism design for multi-robot coordination. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *NIPS*. MIT Press, 2003.

[8] Claude Berge. *The Theory of Graphs*. Dover Books on Mathematics. Dover Publications, 2001.

[9] Daniel S. Bernstein, Shlomo Zilberstein, and Neil Immerman. The complexity of decentralized control of markov decision processes. In Craig Boutilier and Moisés Goldszmidt, editors, *UAI*, pages 32–37. Morgan Kaufmann, 2000.

[10] Dimitri P. Bertsekas. *Dynamic programming and optimal control*, volume 2 of *Optimization and computation series*. Athena Scientific, 2 edition, 2001.

[11] Guido Boella, Jan Broersen, and Leendert van der Torre. Reasoning about constitutive norms, counts-as conditionals, institutions, deadlines and violations. In The Duy Bui, Tuong Vinh Ho, and Quang-Thuy Ha, editors, *PRIMA*, volume 5357 of *Lecture Notes in Computer Science*, pages 86–97. Springer, 2008.

[12] Guido Boella and Leendert van der Torre. A game theoretic approach to contracts in multiagent systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 36(1):68–79, 2006.

[13] Guido Boella and Leendert van der Torre. Substantive and procedural norms in normative multiagent systems. *J. Applied Logic*, 6(2):152–171, 2008.

[14] Guido Boella, Leendert van der Torre, and Harko Verhagen. Introduction to normative multiagent systems. In Boella et al. [15].

[15] Guido Boella, Leendert W. N. van der Torre, and Harko Verhagen, editors. *Normative Multi-agent Systems*, volume 07122 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2007.

[16] Magnus Boman. Norms in Artificial Decision Making. *Artif. Intell. Law*, 7(1):17–35, 1999.

[17] Craig Boutilier. Sequential optimality and coordination in multiagent systems. In Thomas Dean, editor, *IJCAI*, pages 478–485. Morgan Kaufmann, 1999.

[18] Craig Boutilier, Ronen I. Brafman, and Christopher W. Geib. Structured Reachability Analysis for Markov Decision Processes. In Gregory F. Cooper and Serafín Moral, editors, *UAI*, pages 24–32. Morgan Kaufmann, 1998.

[19] Craig Boutilier, Thomas Dean, and Steve Hanks. Decision-Theoretic Planning: Structural Assumptions and Computational Leverage. *J. Artif. Intell. Res. (JAIR)*, 11:1–94, 1999.

[20] Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Exploiting structure in policy construction. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes*, pages 1104–1113. Morgan Kaufmann, 1995.

[21] Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Stochastic dynamic programming with factored representations. *Artif. Intell.*, 121(1–2):49–107, 2000.

[22] Frances M. T. Brazier, Catholijn M. Jonker, and Jan Treur. Compositional Design and Reuse of a Generic Agent Model. *Applied Artificial Intelligence*, 14(5):491–538, 2000.

[23] Jan Broersen, Mehdi Dastani, Joris Hulstijn, Zhisheng Huang, and Leendert van der Torre. The BOID architecture: conflicts between beliefs, obligations, intentions and desires. In *Proceedings of the fifth international conference on Autonomous agents*, Agents '01, pages 9–16, New York, NY, USA, 2001. ACM.

[24] Jan Broersen, Mehdi Dastani, Joris Hulstijn, and Leendert van der Torre. Goal Generation in the BOID Architecture. *Cognitive Science Quarterly Journal*, 2(3-4):428–447, 2002.

[25] Jan Broersen, Frank Dignum, Virginia Dignum, and John-Jules Ch. Meyer. Designing a deontic logic of deadlines. In Alessio Lomuscio and Donald Nute, editors, *DEON*, volume 3065 of *Lecture Notes in Computer Science*, pages 43–56. Springer, 2004.

[26] Henrique Lopes Cardoso and Eugénio C. Oliveira. Adaptive Deterrence Sanctions in a Normative Framework. In *IAT*, pages 36–43. IEEE, 2009.

[27] Ana Casali, Lluis Godo, and Carles Sierra. Graded BDI Models for Agent Architectures. In João Alexandre Leite and Paolo Torroni, editors, *CLIMA V*, volume 3487 of *Lecture Notes in Computer Science*, pages 126–143. Springer, 2004.

[28] Cristiano Castelfranchi, Frank Dignum, Catholijn M. Jonker, and Jan Treur. Deliberative Normative Agents: Principles and Architecture. In Nicholas R. Jennings and Yves Lespérance, editors, *ATAL*, volume 1757 of *Lecture Notes in Computer Science*, pages 364–378. Springer, 1999.

[29] Cristiano Castelfranchi and W. Lewis Johnson, editors. *The First International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2002, July 15-19, 2002, Bologna, Italy, Proceedings*. ACM, 2002.

[30] Ruggiero Cavallo, David C. Parkes, and Satinder Singh. Optimal coordination of loosely-coupled self-interested robots. In *Workshop on Auction Mechanisms for Robot Coordination, AAAI'06*, Boston, MA, 2006.

[31] Ruggiero Cavallo, David C. Parkes, and Satinder P. Singh. Optimal coordinated planning amongst self-interested agents with private state. In *UAI*. AUAI Press, 2006.

[32] Roberto Centeno, Holger Billhardt, Ramón Hermoso, and Sascha Ossowski. Organising mas: a formal model based on organisational mechanisms. In Sung Y. Shin and Sascha Ossowski, editors, *SAC*, pages 740–746. ACM, 2009.

[33] Rosaria Conte and Cristiano Castelfranchi. *Cognitive and social action*. UCL Press, London, 1995.

[34] Natalia Criado, Estefania Argente, and Vicente J. Botti. Open issues for normative multi-agent systems. *AI Communications*, 24(3):233–264, 2011.

[35] Adnan Darwiche and Moisés Goldszmidt. Action networks: A framework for reasoning about actions and change under uncertainty. In Ramon López de Mántaras and David Poole, editors, *UAI*, pages 136–144. Morgan Kaufmann, 1994.

[36] Thomas Dean and Robert Givan. Model minimization in Markov Decision Processes. In Benjamin Kuipers and Bonnie L. Webber, editors, *AAAI/IAAI*, pages 106–111. AAAI Press / The MIT Press, 1997.

[37] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

[38] Thomas Dean and Shieu-Hong Lin. Decomposition techniques for planning in stochastic domains. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes*, pages 1121–1129. Morgan Kaufmann, 1995.

[39] Chrysanthos Dellarocas. Contractual agent societies: Negotiated shared context and social control in open multi-agent systems. In *Social Order in Multi-Agent Systems*, pages 113–133. Kluwer Academic Publishers, 2000.

[40] Pilar Dellunde and Lluis Godo. Introducing grades in deontic logics. In Ron van der Meyden and Leendert van der Torre, editors, *DEON*, volume 5076 of *Lecture Notes in Computer Science*, pages 248–262. Springer, 2008.

[41] Frank Dignum. Autonomous agents and social norms. In *ICMAS'96 Workshop on Norms, Obligations and Conventions, Kyoto*, pages 56–71, 1996.

[42] Frank Dignum, David N. Morley, Liz Sonenberg, and Lawrence Cavedon. Towards socially sophisticated BDI agents. In *ICMAS*, pages 111–118. IEEE Computer Society, 2000.

[43] Edmund H. Durfee. Practically coordinating. *AI Magazine*, 20(1):99–116, 1999.

[44] Marc Esteva. *Electronic Institutions: From Specifications to Development*. PhD thesis, Universitat Politècnica de Catalunya, 2003.

[45] Moser Silva Fagundes, Holger Billhardt, and Sascha Ossowski. Behavior Adaptation in RMAS: An Agent Architecture based on MDPs. In Robert Trappl, editor, *EMCSR*, pages 453–458. Austrian Society for Cybernetic Studies, 2010.

[46] Moser Silva Fagundes, Holger Billhardt, and Sascha Ossowski. Normative Reasoning with an Adaptive Self-interested Agent Model Based on Markov Decision Processes. In Ángel Fernando Kuri Morales and Guillermo Ricardo Simari, editors, *IBERAMIA*, volume 6433 of *Lecture Notes in Computer Science*, pages 274–283. Springer, 2010.

[47] Moser Silva Fagundes, Holger Billhardt, and s Sascha Ossowski. Reasoning about Norm Compliance with Rational Agents. In Helder Coelho, Rudi Studer, and Michael Wooldridge, editors, *ECAI*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 1027–1028. IOS Press, 2010.

[48] Moser Silva Fagundes, Roberto Centeno, Holger Billhardt, and Sascha Ossowski. Designing Organized Multiagent Systems through MDPs. In Lars Braubach, Wiebe van der Hoek, Paolo Petta, and Alexander Pokahr, editors, *MATES*, volume 5774 of *Lecture Notes in Computer Science*, pages 183–188. Springer, 2009.

[49] Moser Silva Fagundes, Sascha Ossowski, Michael Luck, and Simon Miles. Representing and evaluating electronic contracts with Normative Markov Decision Processes. In *AT*. CEUR, 2012.

[50] Moser Silva Fagundes, Sascha Ossowski, Michael Luck, and Simon Miles. Using Normative Markov Decision Processes for evaluating electronic contracts. *AI Communications*, 25(1):1–17, 2012.

[51] William Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, 3 edition, 1968.

[52] Andrés García-Camino, Pablo Noriega, and Juan A. Rodríguez-Aguilar. Implementing norms in electronic institutions. In Frank Dignum, Virginia Dignum, Sven Koenig, Sarit Kraus, Munindar P. Singh, and Michael Wooldridge, editors, *AAMAS*, pages 667–673. ACM, 2005.

[53] Andrés García-Camino, Juan A. Rodríguez-Aguilar, Carles Sierra, and Wamberto Weber Vasconcelos. Constraint rule-based programming of norms for electronic institutions. *Autonomous Agents and Multi-Agent Systems*, 18(1):186–217, 2009.

[54] Claudia V. Goldman and Shlomo Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In Rosenschein et al. [101], pages 137–144.

[55] Georg Gottlob and Toby Walsh, editors. *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*. Morgan Kaufmann, 2003.

[56] Guido Governatori and Zoran Milosevic. A formal analysis of a business contract language. *Int. J. Cooperative Inf. Syst.*, 15(4):659–685, 2006.

[57] Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent planning with factored mdps. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *NIPS*, pages 1523–1530. MIT Press, 2001.

[58] Haipeng Guo and William Hsu. A survey of algorithms for real-time Bayesian network inference. In *AAAI/KDD/UAI-2002 Joint Workshop on Real-Time Decision Support and Diagnosis Systems, Edmonton, Alberta, Canada*, 2002.

[59] Hossam Hanna and Abdel-Illah Mouaddib. Task selection problem under uncertainty as decision-making. In Castelfranchi and Johnson [29], pages 1303–1308.

[60] James E. Hanson and Zoran Milosevic. Conversation-oriented protocols for contract negotiations. In *7th International Enterprise Distributed Object Computing Conference (EDOC 2003), 16-19 September 2003, Brisbane, Australia, Proceedings*, pages 40–49, 2003.

[61] Ronald A. Howard. *Dynamic Programming and Markov Processes*. The M.I.T. Press, 1960.

[62] Michal Jakob, Michal Pechoucek, Simon Miles, and Michael Luck. Case Studies for Contract-based Systems. In *AAMAS Conference Industry Track*, pages 55–62. IFAAMAS, May 2008.

[63] Finn V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2001.

[64] Sindhu Joseph, Carles Sierra, Marco Schorlemmer, and Pilar Dellunde. Deductive Coherence and Norm Adoption. *Logic Journal of the IGPL*, 18(1):118–156, February 2010.

[65] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1–2):99–134, 1998.

[66] John George Kemeny and J. Laurie Snell. *Finite Markov Chains*. Undergraduate texts in mathematics. Springer-Verlag, 1960.

[67] Martin J. Kollingbaum and Timothy J. Norman. Supervised interaction: creating a web of trust for contracting agents in electronic environments. In Castelfranchi and Johnson [29], pages 272–279.

[68] Martin J. Kollingbaum and Timothy J. Norman. NoA - A Normative Agent Architecture. In Gottlob and Walsh [55], pages 1465–1466.

[69] Martin J. Kollingbaum and Timothy J. Norman. Norm adoption and consistency in the noa agent architecture. In Mehdi Dastani, Jürgen Dix, and Amal El Fallah-Seghrouchni, editors, *PROMAS*, volume 3067 of *Lecture Notes in Computer Science*, pages 169–186. Springer, 2003.

[70] Martin J. Kollingbaum and Timothy J. Norman. Norm Adoption in the NoA Agent Architecture. In Rosenschein et al. [101], pages 1038–1039.

[71] Martin J. Kollingbaum and Timothy J. Norman. Informed deliberation during norm-governed practical reasoning. In Olivier Boissier, Julian A. Padget, Virginia Dignum, Gabriela Lindemann, Eric T. Matson, Sascha Ossowski, Jaime Simão Sichman, and Javier Vázquez-Salceda, editors, *AAMAS Workshops*, volume 3913 of *Lecture Notes in Computer Science*, pages 183–197. Springer, 2005.

[72] Steffen Lauritzen and David Spiegelhalter. Local computations with probabilities on graphical structures and their applications to expert systems. *Journal of the Royal Statistical Society, Series B*, 50(2):157–224, 1988.

[73] Victor R. Lesser and Les Gasser, editors. *Proceedings of the First International Conference on Multiagent Systems, June 12-14, 1995, San Francisco, California, USA*. The MIT Press, 1995.

[74] Peter F. Linington, Zoran Milosevic, James B. Cole, Simon Gibson, Sachin Kulkarni, and Stephen W. Neal. A unified behavioural model and a contract language for extended enterprise. *Data Knowl. Eng.*, 51(1):5–29, 2004.

[75] Michael L. Littman, Thomas Dean, and Leslie Pack Kaelbling. On the complexity of solving markov decision problems. In Philippe Besnard and Steve Hanks, editors, *UAI*, pages 394–402. Morgan Kaufmann, 1995.

[76] Michael Luck, Samhar Mahmoud, Felipe Meneguzzi, Martin Kollingbaum, Tim Norman, Natalia Criado, and Moser Silva Fagundes. *Normative Agents*, volume 8 of *Law, Governance and Technology Series*, chapter 14, pages 209–219. Springer, 2012.

[77] Janusz Marecki and Milind Tambe. Planning with continuous resources for agent teams. In Carles Sierra, Cristiano Castelfranchi, Keith S. Decker, and Jaime Simão Sichman, editors, *AAMAS (2)*, pages 1089–1096. IFAAMAS, 2009.

[78] Felipe Meneguzzi, Sanjay Modgil, Nir Oren, Simon Miles, Michael Luck, Noura Faci, Camden Holt, and Malcom Smith. Monitoring and Explanation of Contract Execution: A Case Study in the Aerospace Domain. In *AAMAS Conference Industry Track*, May 2009.

[79] Felipe Rech Meneguzzi and Michael Luck. Composing high-level plans for declarative agent programming. In Matteo Baldoni, Tran Cao Son, M. Birna van Riemsdijk, and Michael Winikoff, editors, *DALT*, volume 4897 of *Lecture Notes in Computer Science*, pages 69–85. Springer, 2007.

[80] Felipe Rech Meneguzzi and Michael Luck. Norm-based behaviour modification in BDI agents. In Sierra et al. [108], pages 177–184.

[81] Zoran Milosevic, Simon Gibson, Peter Linington, James Cole, and Sachin Kulkarni. On design and implementation of a contract monitoring facility. In *Proceedings of the First IEEE International Workshop on Electronic Contracting*, WEC'04, pages 62–70. IEEE Computer Society, 2004.

[82] Sanjay Modgil, Noura Faci, Felipe Rech Meneguzzi, Nir Oren, Simon Miles, and Michael Luck. A framework for monitoring agent-based normative systems. In Sierra et al. [108], pages 153–160.

[83] Ranjit Nair, Milind Tambe, Makoto Yokoo, David V. Pynadath, and Stacy Marsella. Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. In Gottlob and Walsh [55], pages 705–711.

[84] Pablo Noriega. *Agent Mediated Auctions: The Fishmarket Metaphor*. PhD thesis, Universitat Autonoma de Barcelona, 1997.

[85] James Norris. *Markov Chains*. Cambridge University Press, USA, 1999.

[86] Jean Oh, Felipe Meneguzzi, and Katia P. Sycara. Probabilistic plan recognition for intelligent information agents - towards proactive software assistant agents. In Joaquim Filipe and Ana L. N. Fred, editors, *ICAART (2)*, pages 281–287. SciTePress, 2011.

[87] Jean Oh, Felipe Meneguzzi, Katia P. Sycara, and Timothy J. Norman. An agent architecture for prognostic reasoning assistance. In Toby Walsh, editor, *IJCAI*, pages 2513–2518. IJCAI/AAAI, 2011.

[88] Nir Oren, Sofia Panagiotidi, Javier Vázquez-Salceda, Sanjay Modgil, Michael Luck, and Simon Miles. Towards a formalisation of electronic contracting environments. In Jomi Fred Hübner, Eric T. Matson, Olivier Boissier, and Virginia Dignum, editors, *COIN@AAMAS&AAAI*, volume 5428 of *Lecture Notes in Computer Science*, pages 156–171. Springer, 2008.

[89] Sascha Ossowski. *Co-ordination in Artificial Agent Societies, Social Structure and Its Implications for Autonomous Problem-Solving Agents*, volume 1535 of *Lecture Notes in Computer Science*. Springer, 1999.

[90] Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[91] Judea Pearl. Fusion, propagation, and structuring in belief networks. *Artif. Intell.*, 29(3):241–288, 1986.

[92] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. The Morgan and Kaufmann series in representation and reasoning. Morgan Kaufmann, 2 edition, 1988.

[93] Warren B. Powell. *Approximate dynamic programming: solving the curses of dimensionality*. Wiley series in probability and statistics. John Willey & Sons, 2007.

[94] Martin L. Puterman. *Markov Decision Processes – Discrete Schocastic Dynamic Programming*. Wiley series in probability and statistics. John Willey & Sons, 1994.

[95] Martin L. Puterman and Moon Chirl Shin. Modified Policy Iteration Algorithms for Discounted Markov Decision Problems. *Management Science*, 24:1127–1137, 1978.

[96] Omer F. Rana and Kate Stout. What is scalability in multi-agent systems? In *Agents*, pages 56–63, 2000.

[97] Anand S. Rao. AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. In Walter Van de Velde and John W. Perram, editors, *MAAMAW*, volume 1038 of *Lecture Notes in Computer Science*, pages 42–55. Springer, 1996.

[98] Anand S. Rao and Michael P. Georgeff. BDI Agents: From Theory to Practice. In Lesser and Gasser [73], pages 312–319.

[99] Daniel M. Reeves, Michael P. Wellman, and Benjamin N. Grosof. Automated negotiation from declarative contract descriptions. *Computational Intelligence*, 18(4):482–500, 2002.

[100] Joseph F. Ritt. Permutable rational functions. *Transactions of the American Mathematical Society*, 25(3):399–448, 1923.

[101] Jeffrey S. Rosenschein, Tuomas Sandholm, Michael Wooldridge, and Makoto Yokoo, editors. *The Second International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2003, July 14-18, 2003, Melbourne, Victoria, Australia, Proceedings*. ACM, 2003.

[102] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2 edition, 2003.

[103] Mathias Sallé. Electronic contract framework for contractual agents. In Robin Cohen and Bruce Spencer, editors, *Canadian Conference on AI*, volume 2338 of *Lecture Notes in Computer Science*, pages 349–353. Springer, 2002.

[104] Tuomas Sandholm and Victor R. Lesser. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In Lesser and Gasser [73], pages 328–335.

[105] Lloyd S. Shapley. Stochastic games. *PNAS*, 39(10):1095—1100, 1953.

[106] Onn Shehory and Sarit Kraus. Methods for task allocation via agent coalition formation. *Artif. Intell.*, 101(1–2):165–200, 1998.

[107] Yoav Shoham and Moshe Tennenholtz. On social laws for artificial agent societies: Off-line design. *Artif. Intell.*, 73(1–2):231–252, 1995.

[108] Carles Sierra, Cristiano Castelfranchi, Keith S. Decker, and Jaime Simão Sichman, editors. *8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary, May 10-15, 2009, Volume 1*. IFAAMAS, 2009.

[109] Herbert A. Simon. *Models Of Bounded Rationality And Other Topics In Economics: Models Of Bounded Rationality*. The MIT Press, 1982.

[110] Michael Spivey. *Z Notation - A Reference Manual*. Prentice Hall International Series in Computer Science. Prentice Hall, 1992.

[111] Moshe Tennenholtz. On social constraints for rational agents. *Computational Intelligence*, 15(4):367–383, 1999.

[112] Paul Thagard. *Coherence in Thought and Action (Life and Mind: Philosophical Issues in Biology and Psychology)*. The MIT Press, August 2002.

[113] Phillip J. Turner and Nicholas R. Jennings. Improving the scalability of multi-agent systems. In *Revised Papers from the International Workshop on Infrastructure for Multi-Agent Systems: Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems*, pages 246–262. Springer-Verlag, 2001.

[114] Leendert van der Torre and Yao-Hua Tan. Contrary-to-duty reasoning with preference-based dyadic obligations. *Ann. Math. Artif. Intell.*, 27(1-4):49–78, 1999.

[115] Jo van Nunen. A set of successive approximation methods for discounted markovian decision problems. *Z. Operations Research*, 20:203–208, 1976.

[116] Wamberto Weber Vasconcelos, Martin J. Kollingbaum, and Timothy J. Norman. Normative conflict resolution in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 19(2):124–152, 2009.

[117] Javier Vázquez-Salceda, Huib Aldewereld, and Frank Dignum. Implementing norms in multiagent systems. In Gabriela Lindemann, Jörg Denzinger, Ingo J. Timm, and Rainer Unland, editors, *MATES*, volume 3187 of *Lecture Notes in Computer Science*, pages 313–327. Springer, 2004.

[118] Daniel Villatoro, Jordi Sabater-Mir, and Jaime Simão Sichman, editors. *Multi-Agent-Based Simulation XII - International Workshop, MABS 2011, Taipei, Taiwan, May 2-6, 2011, Revised Selected Papers*, volume 7124 of *Lecture Notes in Computer Science*. Springer, 2012.

[119] Michael Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, 2002.

[120] Fabiola López y López. *Social Power and Norms: Impact on agent behaviour*. PhD thesis, University of Southampton, 2003.

[121] Fabiola López y López and Michael Luck. Modelling Norms for Autonomous Agents. In *ENC*, pages 238–245. IEEE Computer Society, 2003.

[122] Fabiola López y López, Michael Luck, and Mark d'Inverno. Constraining autonomy through norms. In Castelfranchi and Johnson [29], pages 674–681.

[123] Fabiola López y López, Michael Luck, and Mark d'Inverno. A Normative Framework for Agent-Based Systems. In Boella et al. [15].

[124] Nevin Zhang and David Poole. A simple approach to Bayesian network computations. In *Proceedings of the Tenth Canadian Conference on Artificial Intelligence*, pages 171–178, 1994.