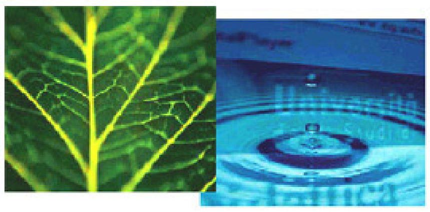
PhD Dissertation



International Doctorate School in Information and Communication Technologies

DISI - University of Trento

Nòmos 3: legal compliance of software requirements

Silvia Ingolfo

Advisor:

Prof. John Mylopoulos

Università degli Studi di Trento

November 2015

Abstract

Laws and regulations are increasingly impacting the design and development of software systems, as legislations around the world attempt to control the impact of software on social and private life. Software systems need to be designed from the beginning in a law-aware fashion to ensure compliance with applicable laws. Moreover, they need to evolve over time as new laws pass and existing ones are amended.

In this interdisciplinary field many challenges remain open. For any given norm, there are alternative ways to comply with it for a system-to-be. Moreover, revising some requirements or adding new ones can have an important impact on what norms apply. To complicate matters, there is a sizeable knowledge gap between technical and legal experts, and this hampers requirements analysts in dealing with the problem on their own. This thesis proposes to use conceptual models of law and requirements to help requirements engineers address these problems by answering questions such as "Given this set of requirements, which norms are applicable?", "Which norms are complied with?", "What are the alternative ways I use to comply with a norm?".

The thesis proposes the Nòmos 3 framework that includes a modeling language for law and requirements, reasoning support for Nòmos 3 models, as well as a systematic process for establishing compliance. The proposed framework is evaluated by means of illustrative case studies, a scalability study for the reasoning mechanism, as well as other specific studies intended to assess the effectiveness of the proposed concepts, tools, and process.

Keywords[Conceptual model, Requirements engineering, Regulatory compliance]

The work compiled in this thesis has been supported by the ERC advanced grant 267856 "Lucretius: Foundations for Software Evolution" (April 2011 – March 2016)

This thesis is dedicated to those I have lost,
and those I have found.
Three years.
Three important persons.

No one knows what he can do till he tries

Acknowledgements

When I started my PhD I honestly had no idea what I was getting myself into.
I thought it was going to be a fun and interesting learning adventure.
The opportunity of an experience right on my fingertip.
A journey so unique and challenging, I could not resist.

A four-year long journey during which I thought I was never going to finish.
A difficult journey I did not always know how to handle.
A journey that took me to places I've never been.
A journey that taught me lessons I never thought I'd learn.

Most importantly, a journey I did not do alone.

It's kind of fun to do the impossible — Walt Disney

In my PhD journey, my advisor Prof. John Mylopoulos has been the admiral of the fleet. The dedication and patience he puts in directing all of us, is simply admirable. I owe him all the lessons learned and all the opportunities I've had during my PhD.

Tell me and I'll forget, teach me and I may remember, involve me and I'll learn — Chinese Proverb

My friends and colleagues have been the other sailors of the fleet. I wont list you all 'cause for sure I would forget some names :-p Sharing these years with you has been fun. Each of you has its own unique *sailing* technique: brazilian, canadian, chinese, albanian, turkish, spanish, romanian or italian. Talking and discussing with all of you — about problems, solutions, or about everything at all — made me a better sailor. I will cherish so many memories. . . of lunches with Carlo. . . of my bus rides and spritzes with Carmen. . . of my roommate (and personal nurse/hospital-ride) Grazia. . . of the incredible parties by Jennifer. . . of my breakfasts and lunches with Tong and Claudia. . . of the conference I've been with some of you and allowed me to know you better. . . or simply of our RE-seminars, from the early ones in Serraia to the lunches in Garda.

A journey is best measured in friends, rather than miles — Tim Cahill

I've crossed paths with incredible professors like Julio ("Siiiiiiiiiilvia, let me tell you something"), Nicola, Ivan, Paolo, Renata, Giancarlo, Oscar, Lysanne and Daniel. It is people like you

— with your passion, your integrity, your knowledge and dedication — who sets high standards for the research community. Alberto, Anna and Angelo: your kindness and experience helped me sailing through problems, deadlines and tough times.

Tough times never last, tough people do — Dr. Robert Schuller

Benny has been the anchor keeping my ship steady when I docked, and the mast of the ship when I sailed. You taught me that no matter what problems and difficult dilemmas you face today, the world would still go around tomorrow (and hopefully tomorrow is Saturday and we can go to the beach! :D) Thank you so much for your support!

Happiness is when what you think, what you say,
and what you do are in harmony — Mahatma Gandhi

Andreza&Giovani and Caroline have been the keel that kept my ship right-side up. Breakfasts and dinners with you are precious memories that I will cherish jealously in my heart. You are really beautiful persons and I'm really glad I've had you next to me in these years.

There is nothing better than a friend, unless it is a friend with chocolate. — Linda Grayson

My family has been the supporter of my journey, pointing ventilators to the sea in all possible directions to help me sail wherever I was going. I wish four years hadn't changed as many things as they had...

Times and conditions change so rapidly that we must keep our aim constantly focused on the future — Walt Disney

Thank you all, thank you so much!

Silvia

Contents

1 Introduction **1**

1.1 The problem of compliance in requirements engineering 1

1.1.1 Challenges of regulatory compliance for requirements engineering 1

1.1.2 The problem of compliance in requirements engineering 4

1.2 Research objectives 6

1.3 Approach overview 8

1.3.1 Contributions 9

1.4 Structure of the thesis 10

1.5 Published work 11

2 State of the art **15**

2.1 Baseline 15

2.1.1 Nòmos 15

2.1.2 Backward forward label propagation algorithms 16

2.1.3 Goal-Oriented Requirements Engineering (GORE) 17

2.1.4 The Business Intelligence Model 19

2.2 Related work 20

2.2.1 Requirements engineering and regulatory compliance 20

2.2.2 Conceptual modeling and law 23

2.3 Related research areas 25

2.3.1 Normative multi-agents systems 25

2.3.2 AI&Law 26

2.3.3 Compliance of business process and enterprise modeling 27

2.3.4 Security and compliance with privacy laws/standards 28

3 Language **31**

3.1 Introduction 31

3.2 Concepts and relationships 32

3.2.1 Concepts 33

3.2.2 Relationships . . . 36
3.2.3 Visual notation . . . 38
3.3 Syntax and semantics . . . 42
3.4 Chapter summary . . . 53

4 Reasoning with variability **55**
4.1 Legal variability . . . 57
4.1.1 Applicability search . . . 57
4.1.2 Applicability analysis . . . 59
4.1.3 Satisfiability search . . . 60
4.1.4 Satisfiability analysis . . . 61
4.1.5 Simple compliance analysis . . . 62
4.1.6 Simple compliance search . . . 64
4.2 Roles fulfillment variability . . . 66
4.2.1 Roles fulfillment search . . . 66
4.2.2 Role fulfillment analysis . . . 68
4.2.3 Analysis of Norms applicable to Legal Roles . . . 70
4.2.4 Analysis of Norms applicable to Social Roles . . . 72
4.3 Conclusions . . . 73

5 Reasoning with violations and goals **77**
5.1 Role-based violations . . . 77
5.1.1 The reserve schema . . . 78
5.1.2 Violations of Norms with the reserve schema . . . 80
5.1.3 Legally-valid social delegation . . . 87
5.2 Goal model compliance . . . 88
5.2.1 Goal model compliance analysis . . . 89
5.2.2 Goal model compliance search . . . 91
5.3 Conclusions . . . 95

6 Systematic process for revising requirements **97**
6.1 Overview of the process . . . 97
6.1.1 Compliance target . . . 99
6.2 The compliance process . . . 100
6.2.1 Analysis of the model . . . 100
6.2.2 Compliance evaluation . . . 102
6.2.3 Model revision . . . 103
6.3 Conclusions . . . 107

7 Evaluation of the Nòmos 3 language and reasoning **109**
7.1 An analysis of the Italian Guidelines on EHR with Nòmos 109
7.1.1 Analysis method 110
7.1.2 Example of analysis 111
7.1.3 Results 111
7.1.4 Discussion 117
7.2 Evaluation with illustrative example 118
7.2.1 Reasoning with legal variability 120
7.2.2 Reasoning with role variability 124
7.2.3 Reasoning with compliance of a goal model 126
7.3 Conclusions 129

8 Evaluation of the scalability **131**
8.1 Automated reasoning 131
8.2 Scalability of the approach 142
8.2.1 Design of the experiment 143
8.2.2 Experiment results 144
8.2.3 Discussion and limitations 146
8.3 Conclusions 147

9 Evaluation of the process **149**
9.1 Initial evaluation of the systematic process 149
9.1.1 Design 150
9.1.2 Execution of the case study 151
9.1.3 Results 151
9.2 Systematic process: an illustrative example 153
9.2.1 Analysis phase 154
9.2.2 Compliance evaluation phase 154
9.2.3 Revision phase 156
9.3 Conclusions 161

10 Conclusions **163**
10.1 Contributions to the state-of-the-art 163
10.2 Ongoing work and open issues 165
10.2.1 Preferred compliance problem 165
10.2.2 Regulatory compliance in adaptive systems 166
10.2.3 Nòmos 2 for compliance of Business Processes 169
10.3 Limitations and future directions 171

Appendix A NRTool configuration file **173**

List of Tables

3.1 Conjunction and disjunction for satisfaction values. 43
3.2 Rules for assigning values to Norms. 44
3.3 Conjunction and disjunction of fulfillment values. 46
3.4 Propagation rules for $satisfy/satisfy^*$ and $break/break^*$ relations. 47
3.5 Propagation rules for *holder*, *wanted* and *coplay* relations. 49

4.1 Algorithms for reasoning about the variability of law in a Nòmos 3 model. 74
4.2 Algorithms for reasoning about the variability of Role fulfillment in a Nòmos 3 model. 75

5.1 The truth table for the formula representing the reserve schema. 81
5.2 Case 1: the reserve schema applies and both conditions are met. 81
5.3 Case 2: the reserve schema applies and condition (a) is not met. 81
5.4 Case 3: the reserve schema applies and condition (b) is not met. 83
5.5 Case 4: the reserve schema applies and neither conditions are met. 85
5.6 Case 5: the reserve schema does not apply. 86

6.1 Possible combinations of compliance value: target values vs values obtained in the analysis. 99

7.1 Numerical results of the analysis of the Italian Guidelines on Electronic Health Records with Nòmos [Siena, 2010] 112

9.1 The numerical results of the pilot case study. The adequacy of the framework concepts and process has been validated against two measures: one from the transcripts of the session recordings, and one from post-study interview of the participants. 151
9.2 Comparison of the compliance values for the Norms in the the example of a software for online sellers. 157

List of Figures

1.1 Characterization of the type of requirements that come from stakeholders R, from the law L, and that "overlaps" between the previous two ($R \cap L$). 5

3.1 Relationships of the Nòmos 3 language. 36
3.2 Visual notation for the modelling language. 39
3.3 Class diagram of the Nòmos 3 concepts. 40
3.4 BNF rules defining the grammar of the Nòmos 3 language. 52

4.1 Applicability search/analysis: scenario of the online seller. 58
4.2 Satisfiability search/analysis: scenario of the online seller. 60
4.3 Simple compliance search/analysis: scenario of the online seller. 63
4.4 Role fulfillment search: scenario of the online seller. 67
4.5 Role fulfillment analysis: scenario of the online seller. 69
4.6 Noms applicable to legal role analysis: scenario of the online seller. 71
4.7 Noms applicable to legal role analysis: scenario of the online seller. 73

5.1 The concepts and relationships defining the reserve schema. 79
5.2 Reserve schema: example. 80
5.3 Schema violation 2a. 82
5.4 Schema violation 2b. 82
5.5 Schema violation 2c. 83
5.6 Schema violation 3a. 84
5.7 Schema violation 3b. 84
5.8 Schema violation 3c. 85
5.9 Schema not applicable. 86
5.10 Example of Social Delegation in Nòmos 3. 87
5.11 Example of two social delegation: only one is legally valid. 88
5.12 Full compliance analysis: scenario of the online seller. 90
5.13 Full compliance search: scenario of the online seller. 94

6.1 Workflow of the proposed compliance process 98

6.2 Workflow detailing the activities in the analysis phase of the compliance process. 101
6.3 Workflow detailing the activities in the compliance check of the compliance process. 102
6.4 Workflow detailing the activities in the revision phase of the compliance process. 104
6.5 Workflow detailing the activities of choosing and applying a revision strategy in the revision phase. 105

7.1 Example of a Nòmos 3 model — including both legal and requirements model — for tax and customs for goods shipped to U.K. in the example of a software for online sellers. 119
7.2 The Nòmos 3 legal model for the running example of tax and customs for goods sent to U.K. from abroad: applicability-analysis() 120
7.3 The Nòmos 3 legal model for the running example of tax and customs for goods sent to U.K. from abroad: compliance-search() 123
7.4 Example of a Nòmos 3 legal model — with Legal Roles — for tax and customs for goods sent to U.K. from abroad: role-fulfill-analysis() 125
7.5 The Nòmos 3 requirements model for the online seller example, evaluated with the current scenario. 127
7.6 The evaluated Nòmos 3 legal model for the online seller example, evaluated with the current scenario. 128
7.7 The evaluation of the reserve schema in the Nòmos 3 model for the online seller example. 128

8.1 The NRTool transforms the Nòmos 3 model provided by the analyst into a disjunctive Datalog program, and reports the output of the Datalog engine back to the analyst. 132
8.2 An example of the input file that the NRTool translates in Datalog. 133
8.3 An example of the output of the tool. 134
8.4 The three steps used by the generation-algorithm to generate artificial Nòmos 2 legal models. 143
8.5 Results from the first scalability experiments analyzing the size of the problem. . 145
8.6 Results from the second scalability experiments analyzing the complexity of the problem. 146
8.7 Results from the third scalability experiment analyzing the space of alternatives. 147

9.1 The Nòmos 3 model for the example of a software for online sellers to help evaluate taxes and documents for shipping goods to U.K. 153
9.2 The applicability analysis Nòmos 3 model for the example of a software for online sellers to help evaluate taxes and documents for shipping goods to U.K. 155

9.3 The compliance evaluation phase of the example of a software for online sellers to help evaluate taxes and documents for shipping goods to U.K. 156
9.4 The first revision of the model resolves the violation on Norm n_5 that was triggered by the reserve schema. 157
9.5 The second revision of the model resolves the violation on Norm n_3. 158
9.6 The second revision of the model resolves the violation on Norm n_1. 159
9.7 The Nòmos 3 legal model and requirements model with all corrections implemented. 160
9.8 The final Nòmos 3 requirements model compliant with all applicable laws. 160

10.1 LAwReq examples elicited based on the autonomous car model [?]. 167
10.2 Reasoning with a Nòmos 2 legal model in URN [Ghanavati et al., 2014b]. 170

Chapter 1

Introduction

In the last two decades we have witnessed an exponential growth in technology that has penetrated all aspects of human activity. In private lifes smartphones, laptops and tablets now allow us to use an app for nearly anything: from tracking our running route, to monitoring our bank account, to booking a flight. In the business world, technology allows now instantaneous file backups in the cloud, or real time monitoring of data streams, from stock markets to a patient's heart beats. And as technology advances, society adapts and tries to keep up. At the same time, laws have to keep up with these advances, circumscribing the limits of technological intrusion. Nowadays any software company or private developer who wants to build a new iPhone app, a web-service, or a computer software, has to check current legislation for possible limitations or compulsory features that the software must have in order to ensure compliance with the law.

In this chapter, we discuss the issue of evaluating compliance of a software system from a requirements engineering perspective, i.e. the very beginning of a software development where it is defined what the software will do. At the end of the chapter we present an overview of the main objective of this thesis: a lightweight framework for evaluating the legal compliance of software requirements.

1.1 The problem of compliance in requirements engineering

1.1.1 Challenges of regulatory compliance for requirements engineering

The problem of compliance in requirements and software engineering has mainly three peculiarities: it is pervasive, expensive to solve, interdisciplinary, and evolving.

Pervasive. The problem of complying with regulation has virtually no border, as there is no bound on the new areas that will be regulated in the future. Every day technology rapidly advances in so many fields that it is difficult for society to keep up with the direction that these advances should take.

Think of this drone technology and the need to regulate how these devices may impact privacy

(“Small Drones Deserve Sensible Regulation: Sensible regulation of small drones would foster innovation and protect privacy”[1]). The need of privacy regulation in this emerging field is clear when considering that this technology can produce surveillance drones that look like insects or hummingbirds (“Could Domestic Surveillance Drones Spur Tougher Privacy Laws?”[2]). Similarly, unmanned-vehicles, or even the new software running on many cars that allows the vehicle to self-drive, self-break and self-change-lane, are clear examples of how technology is advancing and how much society needs to catch up in regulating these technologies. Recent developments in nanotechnology have called for regulations also in the development of nanomaterials in food, drugs and even cosmetics (“US Government Regulators Take on Nanomaterials”[3]).

So any software developed to support any and more of these technology-intensive domain will need to face sooner or later the step evaluating whether the norms applicable are respected.

Expensive. The cost of complying with regulations can be really high and several studies have analyzed the impact of compliance in businesses. It has been estimated that in the Healthcare domain alone, organizations have spent $17.6 billion over a number of years to align their systems and procedures with the Health Insurance Portability and Accountability Act (HIPAA), introduced in 19961. In the Business domain, it was estimated that organizations spent $5.8 billion in one year (2005) to ensure compliance of their reporting and risk management procedures with the Sarbanes-Oxley Act (SOX)2. However, an equally important evaluation is to evaluate the extent of the costs of non-compliance. One of the most thorough investigation in the costs of compliance and non compliance has been carried out by Ponemon in 2011.[4] Their study clearly shows how the cost of non-compliance is more expensive than investing in compliance activities: while the average cost for compliance is more than $3.5 million, the average costs of non-compliance is nearly $9.4 million. The latter is interestingly further analyzed in terms of its consequences: non-compliance leads not only to fines and penalties, but also to business disruption, productivity/revenue loss and settlements costs.

Thompson Reuters GRC-team performed surveys in 2008, 2010 and 2011 with several hundreds of compliance practitioners in order to evaluate the cost of compliance.[5] Their analysis was more focused on understanding the time spent by the compliance team. The results show how the focus will be not in the compliance function itself, but in handling the regulatory information from and with regulators. From a financial point of view, the survey also shows that “*the*

[1]`http://spectrum.ieee.org/aerospace/aviation/small-drones-deserve-sensible-regulation`, accessed on May 2015.

[2]`http://spectrum.ieee.org/automaton/robotics/military-robots/could-domestic-surveillance-drones-spur-tougher-privacy-laws`, accessed on May 2015

[3]`http://spectrum.ieee.org/nanoclast/at-work/innovation/us-government-regulators-take-on-nanomaterials`, accessed on May 2015.

[4]`http://www.ponemon.org/library/the-true-cost-of-compliance-a-benchmark-study-of-multinational-organizations`, accessed on May 2015.

[5]`http://accelus.thomsonreuters.com/special-report/cost-compliance-survey-2012`, accessed on May 2015.

majority of firms expect the cost of compliance resources to increase in 2012". Consequently it is pointed out the increasing important of the role of a compliance officer also in the planning of resources and budget. Among the many challenges, "*the issues most often highlighted by firms for 2012 were keeping on top of regulatory change and the perennial issue of resources*".

Very recently, a report from July 2014 shows and analyzes the increasing costs of compliance for small businesses in UK.[6] Also this report confirms that compliance will have a cost to business of £0.9bn. "*In total this accounts for a 4% increase in the estimated cost of compliance from £18.2 billion to £19.1 billion, with external costs rising to £7.4 billion and internal cost rising to £11.7 billion*".

To the best of our knowledge, there has been no similar survey/analysis of compliance costs for software companies. Some analysis of software licensing compliance have however been carried out,[7] and confirms that "*there are very real costs for poor software license governance and manufacturer audits*" and the detailed type of costs are similar and comparable with the categories identified by Ponemon.

Interdisciplinarity. No software developer in any field can safely assume that no regulation will ever impact his code, and no software company can ignore or postpone the problem of evaluating compliance if they don't want to be out of business. The question is now: who should evaluate whether the software complies with a given piece of law? The interdisciplinarity of the problem is such that software engineers/developers do not have a legal background such that you can rely on their understanding of what the law says. Legal experts do not have the technical background to understand the software details that can be important to define and understand its compliance.

Theoretically in this field, the best candidate to evaluate compliance is somebody with a strong understanding of law and a strong understanding of what the software does and how it does it.[8] Always theoretically, this lawyer-developer candidate is involved at any step of the software development. This way the final product has been produced with a theoretical 100% legal compliance.

Practically — until these lawyer-developer candidates become more available — every company deals with the problem in many different ways: some just have lawyers go through the software specification, some consult with lawyers at the early stage of SW development and then the developer proceeds with the comments from lawyers, lawyer are consulted at every step (but

[6] `https://www.fpb.org/press/july-2014/cost-compliance-continues-rise-small-firms-forum-research-shows`, accessed on May 2015.

[7] `http://blog.shi.com/2013/06/10/the-real-costs-of-being-out-of-compliance-with-your-software-licenses/`, accessed on May 2015.

[8] From an educational point of view where are some attempts in creating these 'special' computer scientists with at least some legal background by including dedicated courses in the studies (for example the "Privacy Technology, Policy, and Law" class at Georgia Tech). Also worth mentioning is the "PhD in Law, Science and Technology" (`http://www.unibo.it/en/teaching/phd/2014-2015/law-science-and-technology-1`) that a group of University are offering indeed to address this interdisciplinary candidate.

increasing the overall costs of the process). No matter the solution, after the interaction between the developer and lawyer happens, the software development/developer continues. Necessarily the theoretical legal compliance achieved now is lower than the case before.

The interdisciplinarity of the problem poses therefore important challenges on how and who should deal with the problem. The project managers are then challenged to include in their fixed budged, a plan that includes methodologies and solutions that support the compliance problem from the early stage of the software development, and therefore support also the analysts.

Evolution. The juridical systems of today are actually the result of hundreds of years of legal evolution. An important characteristic of the body of laws governing a country, is that they must be "constantly" updated or integrated in order to regulate and deal with the problems in the current society. So when at time t_0 a law is passed and later on a software is developed in compliance with that law, it is necessary that compliance is checked at any time of the life of that software. It is in the evolving nature of law that at any time t_1 a new law may be introduced or the previous one is amended, and the changes can deeply affect the software behaviour. For example very recently (as of June 2, 2015) the Italian law mandates that all sites that use in some way cookies insert a banner to alert users.[9] So even just a simple design choice as the one of using cookies, may trigger in the future applicable laws. So the evolution of laws and results in the evolution of software systems. It is therefore important that any software is designed in compliance with the law, and is maintained compliant throughout its lifetime.

1.1.2 The problem of compliance in requirements engineering

The problem of establishing the compliance of software requirements with some law is not trivial and has many challenges. In order to move towards more compliance-oriented software development — or at least compliance-aware development — it is important to start from the beginning of software development and support analysts/developers to achieve compliance already during the early stage of requirements engineering: compliance-aware requirements engineering.

The "traditional" problem in requirements engineering amounts to analyzing the problem that some stakeholders have, and evaluate what the software should do (and how) in order to address the problem of the stakeholders. Requirements engineering is about eliciting and analyzing stakeholder needs in order to produce a specification for the system-to-be. Using the formulation proposed by Zave and Jackson [1997], the requirements problem can be formulated as follows: Given domain assumptions D and requirements R, find a design of the system-to-be, such that its specification S is consistent with D and R, and together with D satisfies R, i.e.,

$$D \cup S \vdash R$$

where $\vdash$ is the consequence relation of classical logic.

[9] http://www.garanteprivacy.it/cookie.

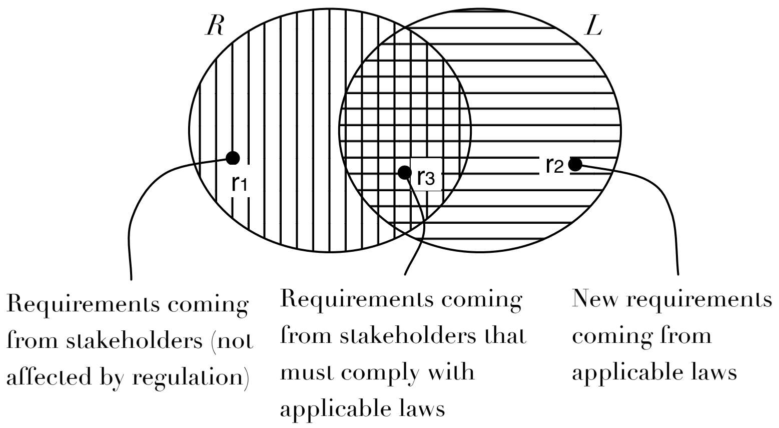

R
L
r1
r2
r3
Requirements coming from stakeholders (not affected by regulation)
Requirements coming from stakeholders that must comply with applicable laws
New requirements coming from applicable laws

Figure 1.1: Characterization of the type of requirements that come from stakeholders R, from the law L, and that "overlaps" between the previous two ($R \cap L$).

However laws and regulations also dictate what the software must and must not do. So the stakeholders describe what is the problem and what they *would like* the software to do.[10] Requirements engineering must then revise the requirements (R') in order to accommodate what the law allows/mandates the software to do. In terms of the Zave and Jackson formulation, we therefore must account for the fact that the specification must be consistent not only with D and R, but also with what the applicable law mandate the software to do L:

$$D \cup S \vdash R', \quad R' = R \cup L\,,$$

where now the specifications are consistent with the revised version of the requirements R' that takes into account the applicable laws L.

The set of what the stakeholders want the software to do R, and what the law wants the software to do L are not disjoint: indeed the problem in requirements engineering is to also include what the law wants in the software specifications and align it with the requirements. In the following we use set-theory to characterize the three cases in $R \cup L$, and use an example of a software for sharing files online.

- $r_1 \in R$: r_1 is a requirement coming from the stakeholder and that is not affected by regulation. For example, "The user shall be able to add a profile picture to his/her profile".
- $r_2 \in L$: r_2 is a requirement that needs to be added to the original set R to achieve compliance with applicable law. For example, "The user personal data that are stored in the server must be anonymized".

[10]More precisely, the activities of identifying the problem and the requirements are elicited from the stakeholders.

- $r_3 \in (R \cap L)$: r_3 is a requirement that comes from the stakeholders but conflicts with the law. r_3 needs to be revised to ensure compliance with the law. In the intersection $R \cap L$ we have two cases:
  - r_3' is a requirement from the stakeholder that is compliant with the law: $r_3' \vdash L$. For example "the software shall allow the user to share a playlist" is compliant with applicable regulation on copyright.
  - r_3'' is is a requirement from the stakeholder that is not compliant with the law: $r_3'' \nvdash L$. For example "the software shall allow the user to share music files" is not compliant with the applicable regulation on copyright.

1.2 Research objectives

Traditional RE methodologies have been developed to solve the traditional $D \cup S \vdash R$ problem, and they do not natively support regulatory compliance (see for example van Lamsweerde [2004]; Yu [1997a]; Bresciani et al. [2004]). Several solutions have been proposed to help dealing with the requirements in $R \cap L$ and L (see section 2.2.1), however most of the proposed solutions are either tailored to one specific law, or rely on an underlying assumption that the 'simple' operationalization in the requirements of the legal text is enough to ensure compliance. Moreover sometimes companies are willing to make compromises between implementing changes and violating some laws.[11] So compliance is not necessarily a binary function but rather a more flexible concept.

In order to support the problem of regulatory compliance and its important challenges, it is important that a solution to the problem addresses these limitations:

- *Domain independence*: a solution to the problem of designing software requirements that complies with all applicable laws.
- *Domain variability*: a solution addressing variability of legal texts, therefore a solution that takes into account the variability of possible laws applying, and of the different ways for complying with it.
- *Flexible compliance*: a solution that takes into consideration the trade-offs that a company may be willing to make between what the law wants, and want the software requirements are.

[11]Microsoft and Samsung are example of big companies which prefer to violate some laws (or at least file gigantic lawsuits) rather than changing their product (see http://www.forbes.com/sites/amitchowdhry/2014/08/06/apple-and-samsung-drop-patent-disputes-against-each-other-outside-of-the-u-s/). For years actually Microsoft kept on releasing new versions of its operating system which were (allegedly) infringing Apple's copyright on the interface elements (see http://www.zdnet.com/article/30-years-before-samsung-when-apple-sued-microsoft/).

- *Knowledge gap*: a solution acknowledging the knowledge gap between the technical background of the analyst and the legal/domain knowledge of legal experts and stakeholders

The research objective of this thesis is to develop a framework for establishing compliance of software requirementswith a set of norms.

The following research questions are answered in addressing our objective:

RQ1: How can we represent both law and requirements? The representation of the variability underlying laws and requirements is an essential property for the ability to reason about compliance. So on one side it is necessary to represent the legal variability, and on the other to link this variability to that of requirements: different requirements make different laws applicable or compliant. Modeling this variability allows to perform compliance reasoning needed to design a compliant set or requirements. We address this question by proposing a modeling language for laws and requirements called Nòmos 3 which allows the analyst to reason about compliance of requirements. The proposed language is propositional and is based on a small set of concepts and relationships, which make the modeling language very lightweight from a computational and modelling standpoint. The flexibility of the language allows to reason over the variability surrounding a single norm's applicability, compliance, or violation, as well as overall compliance/violation of a whole piece of law.

RQ2: How to reason about compliance of a set of requirements? For software requirements, complying with a law depends largely on two factors: what is done/what can be done, and by whom. For example: is a music file been shared or copied? if so, is it shared by the copyright owner or by somebody else? Similarly, is a medical report being modified/created or is it only accessed for reading? by whom? Being able to reason about requirements compliance amounts to reason over the variability of what must be done to comply with a norm, and evaluate who is entitled to do so. We address this question by proposing a set of algorithms for reasoning about the different aspects that characterize a norms compliance. On one side the semantics of the proposed Nòmos 3 language allows to explore the space of solutions of a model and evaluate — through propagation algorithms — the different ways for complying with a norm. On the other side, the syntax of the language allows us — through a schema of relationships — to identify violations of a norm that arises when in the requirements the wrong role wants to achieve an objective (e.g. when it is not the Doctor the one who wants to modify a patient medical record, but for example the Patient).

RQ3: How can we facilitate the use of the Nòmos 3 framework? One of the most important challenges in this area, is to address the knowledge gap that there is between the analysts and the legal experts. We address this question by suggesting a systematic process that guides the analyst from an initial set of requirements to a revised version that meets the desired compliance level. The analyst will use the Nòmos 3 framework to answer important questions about the requirements, applicable and compliant norms. We propose some reasoning algorithms that will help and guide the analyst through the revisions of the requirements model, and legal expert will be involved in the loop only when some specific issues arise.

RQ4: How well does the proposed framework performs when applied to realistic settings? Validation is an important aspect of any research proposal. In this thesis we used three evaluation methods to validate our proposal: scenarios (descriptive method), simulation (experimental method), case study (observational method) [Hevner et al., 2004]. We use different scenarios to illustrate the different aspects of our proposal and to highlight the features of our framework. In order to bring the framework closer to practices, we performed some scalability experiments with large artificial models of law reporting on the scalability of our approach with real-size laws. Moreover, we report on the results of the analysis of an Italian Law (Italian Guidelines on Electronic Health Record and Health File) as a preliminary evaluation of the concepts needed by the language for modeling law. Lastly, descriptive case studies are used to evaluate and illustrate the approach and technique proposed in this thesis.

In the following section we summarize our approach in order to evaluate how it contributes to answering our research questions.

1.3 Approach overview

Modeling approaches intended for law have been studied for decades in AI [Boella et al., 2009; Bench-Capon et al., 2012] and they are generally grounded on expressive, often deontic, logics. However, the type of analysis and reasoning that the requirements engineers need when tackling the problem of regulatory compliance is however at a different level of granularity where a heavy-handed logical representations does not properly support them. For this reason this thesis we propose a conceptual modelling approach to address the problem and help the requirements engineer answer and reason over specific questions about the requirements model at hand.

The proposed modeling language called Nòmos 3, supports through a minimal set of concepts and relationships, the most important features needed to model the law and requirements, and reason about them. Nòmos 3 models are made of typed constants, each referring to a statement in unrestricted natural language, and a predefined set of relationships to indicate interactions between the statements. The basic building blocks of Nòmos 3 models are situations — propositions used to describe a partial state of the world — and the relationships characterizing their interaction with the norms representing the law, the goals representing the requirements, and the roles in both law and requirements. The relationships in the model indicate how the satisfaction of some elements influences that of others. Propagation algorithms and schemas of relationships in the model, are the two fundamental mechanism underlying the reasoning algorithms that allows the analyst answer important questions about the model. For example, given a set of requirements: what are the norms applicable? what are the alternative ways to comply with a norm? are there any corrections that needs to be implemented in the requirements model to achieve the desired compliance target? if a norm is violated, what needs to be done to solve the violation? All these and other questions are structured and organized within a systematic process that guides the analyst from an initial set of requirements, to their revised version that meets the desired compliance with the law. The process is articulated in mainly three phases — analysis, evaluation, and revision of the model — which are repeated until a satisfactory compliance of the requirements model is achieved. The analysis step is used to identify and reason about the laws applicable to the initial set of requirements, and the evaluation step is used to check whether the desired compliance has been achieved. The revision steps guides the analyst through the revision of the model by suggesting alternative ways on how to comply with the violated norms, and how to amend the model in order to solve these violations.

1.3.1 Contributions

In summary the contributions of the thesis are the following:

- A lightweight modeling language for representing law and requirements. This contribution addresses research question **RQ1**.

- A set of algorithms for reasoning about the variability of law and requirements. Reasoning about the variability of law involves understanding why some laws require the software to respect some norms, how the software complies with/violate these norms, and how different laws influence each other. The intersection of this legal variability with the requirements variability allows the analyst to reason about how different sets of requirements make different laws complied or violated. This contribution addresses research questions **RQ2** and **RQ3**.

- A set of algorithms for reasoning about how the roles in both domains influence compliance. The goal of this contribution is to reason and assess the impact of roles in the legal and requirements domain on achieving compliance. This contribution addresses research questions **RQ2** and **RQ3**.

- A set of algorithms for reasoning about violations of norms and overall compliance of a set of requirements. The aim of these algorithms is to help the analyst resolve a legal violation and identify how to achieve compliance of the requirements. This contribution addresses research questions **RQ2** and **RQ3**.

- A systematic process to help the analyst evaluate the compliance of a set of requirements with a given law. This contribution addresses research question **RQ3**.
- An evaluation of the modeling language, reasoning capabilities, and systematic process. This contribution addresses research question **RQ4**.
- An implementation of the modeling language in a command-line tool for the analysis of Nòmos 3 models. This work contributes to research questions **RQ2**, **RQ3** and **RQ4**.

An important characterization and fundamental difference from our approach and the other approaches in the state-of-the-art, is the fact that the concept of “compliance” of a set of requirements can be tailored to meet the needs of the stakeholder. In our proposal we want to allow a company to make such tailored decisions about compliance and support it in the design of the software requirements. Another important characterization of our approach is the ability to reason over models of the law to help the analyst understand the impact of a requirement (or its amendment) to its compliance with the law. Modeling the different relationships between norms, allows our modeling language and process to accommodate amendments, integrations or expansions to the body of laws, making the overall proposal flexible to the evolution of the laws.

1.4 Structure of the thesis

To present the approach summarized in the previous section, the remainder of the thesis is structured as follows:

- Chapter 2 summarizes the state-of-the-art related to our work. First, the baseline for our proposal, including Nòmos, backward and forward label propagation algorithms, Goal-Oriented Software Engineering (GORE) and BIM. Then we examine other approaches for evaluating the legal compliance of requirements, and review some related work in other research areas.
- Chapter 3 presents the Nòmos 3 modeling language. We provide initially an overview of the concepts and relationships that are at the basis of our conceptual model. Then we provide a detail description of the syntax and semantics of the language.
- Chapter 4 introduces the different type of variability reasoning that are possible with the Nòmos 3 modeling language and the questions that can be answered in our models. Different algorithms are proposed and illustrated with a simple running example.
- Chapter 5 introduces a mechanism for identifying violations of norms in our models through schemas of relationships. Then we proposed and illustrate the two reasoning algorithms for evaluating the compliance of a set of requirements.

- Chapter 6 presents the systematic process to help the analyst evaluate the compliance of a set of requirements and amend them when needed. First we provide an overview of the process and its steps, and then we describe in more detail the three parts of the compliance process.

- Chapter 7 shows a preliminary evaluation of the language and its concepts. First it describe the analysis of the Italian guidelines on Electronic Health Record that was done at the beginning of the research, in order to evaluate and identify the characteristics and concepts needed in the modeling language. Secondly an illustrative example is used to perform a preliminary evaluation of the proposed Nòmos 3 language.

- Chapter 8 presents the results from a scalability study to evaluate the robustness of our approach to real-size laws. First we introduce the tool that implements the primitives of the language and allows the user to perform automated reasoning over Nòmos 3 models. Then we report on the experiments performed to evaluate the scalability of our approach.

- Chapter 9 presents an evaluation of the process presented in chapter 6. First we report on a case study done at the beginning of the research, used to evaluate and identify the important steps needed in the compliance process. Then we illustrate the process through an illustrative example.

- Chapter 10 concludes the thesis with a summary of of open issues and ongoing-work that is under investigation, limitations of our approach, and the future directions opened by our research.

1.5 Published work

Journals

- S. Ingolfo, A. Siena, J. Mylopoulos, A. Susi, A. Perini, "Arguing Regulatory Compliance of Software Requirements", Data & Knowledge Engineering (DKE), Volume 87, September 2013, Pages 279-296 (Special issue on ER 2011).

International Conferences

- S. Ingolfo, I. Jureta, A. Siena, A. Perini, A. Susi, J. Mylopoulos, "Nòmos 3: Legal Compliance of Roles and Requirements", 33rd International Conference on Conceptual Modeling - (ER 2014), Atlanta, USA. October 2014.

- A. Siena, S. Ingolfo, A. Perini, A. Susi, J. Mylopoulos, "Automated Reasoning for Regulatory Compliance", 32nd International Conference on Conceptual Modeling - (ER 2013), Honk Kong, China. November 2013.

- S. Ingolfo, A. Siena, I. Jureta, A. Susi, A. Perini, J. Mylopoulos, "Choosing Compliance Solutions Through Stakeholder Preferences", 19th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2013), Essen, Germany. April 2013.

- A. Siena, I. Jureta, S. Ingolfo, A. Susi, A. Perini, J. Mylopoulos, "Capturing Variability of Law with Nòmos 2", 31st International Conference on Conceptual Modeling (ER 2012), Florence, Italy. October 2012. Acceptance rate 17%, 176 submissions

- S. Ingolfo, A. Siena, J. Mylopoulos, "Establishing Regulatory Compliance for Software Requirements", 30th International Conference on Conceptual Modeling (ER 2011), Brussels, Belgium. November 2011. Acceptance rate 15.7%, 157 submissions (Ranked among top 5 articles of the conference)

International Working Conference/Symposium

- S. Ghanavati, S. Ingolfo, A. Siena, "Exploring Legal Business Process Paths", 9th IFIP WG 8.1 working conference on the Practice of Enterprise Modelling (PoEM 2014), Manchester, UK. November 2014.

- S. Ingolfo, V. Souza, "Law and Adaptivity in Requirements Engineering", 18th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2013), San Francisco, U.S.A. May 2013.

Poster/Demo

- S. Ingolfo, A. Siena, J. Mylopoulos, "Nòmos 3: reasoning about regulatory compliance of requirements", RE14, Poster and Demo session.

International Workshop

- S. Ingolfo, A. Siena, J. Mylopoulos, "Modeling Regulatory Compliance in Requirements Engineering", First International Workshop on Conceptual Modeling in Requirements and Business Analysis (MReBA), Atlanta, U.S.A. October 2014.

- S. Ingolfo, A. Siena, J. Mylopoulos, "Goals and Compliance in NÃšmos 3", 7th International i* Workshop (iStar'14), Thessaloniki Greece. June 2014.

- S. Ingolfo, A. Siena, A. Perini, A. Susi, J. Mylopoulos, "Modeling Laws with Nòmos 2", 6th International Workshop on RE and LAW: Special Track on Convergent Challenges, Rio de Janeiro. July 2013.

- S. Ingolfo, J. Horkoff, J. Mylopoulos, "Uncertainty in Goal and Law Modeling and Analysis", 6th International i* Workshop (iStar'13), Valencia Spain. June 2013.

- A. Siena, S. Ingolfo, A. Susi, I. Jureta, A. Perini, J. Mylopoulos, "Requirements, Intentions, Goals and Applicable Norms", Workshop on Requirements, Intentions, and Goals in Conceptual Modeling (RIGIM 2012), Florence, Italy. October 2012.

- S. Ingolfo, A. Siena, J. Mylopoulos, A. Perini, A. Susi, "Nòmos: from Strategic Dependencies to Obligations", 5th International i* Workshop (iStar'11) pg. 72-77, Trento Italy. August 2011.

Chapter 2

State of the art

In this chapter we summarize the state-of-the-art in our area of research. First we introduce the baseline of our work, providing an overview of the works on which our research relies on. Then we provide an overview of the main works in the state of the art that are related to our proposal. Lastly — since the problem of compliance with the law is an issue affecting multiple domains other than requirements engineering — we will provide a brief overview of the main works about compliance in some related research areas.

2.1 Baseline

2.1.1 Nòmos

Nòmos [Siena, 2010] is a modeling framework that enables requirements analysis to deal with the issue of requirements compliance. Its goal-oriented solution is integrated with elements of legal theory so that the representation and modeling includes the elements needed to describe compliance requirements. The Nòmos framework was developed as an extension of i* [Yu, 1997a], which otherwise would have not allowed to express legal concepts. Nòmos adopts the Hohfeldian taxonomy [Hohfeld, 1917], a milestone of juridical literature that proposes a classification of legal concepts that describe the legal relationships between the actors.

Nòmos expands i* by introducing the concept of Normative Proposition together relationships for relating it with the intentional elements (IE) in the i* model. The Normative Proposition (NP) is a concept describing the most atomic information expressing a legal prescription. It basically embodies the basic element of the Hohfeldian concept of rights. As the notion of rights involves two subjects — the right holder and its correlative — it generates different relationships between the two depending on which right it represents: privilege-Noclaim, claim-Duty, power-Liability, or immunity-Disability relationship.

As Nòmos combines elements from both goal-oriented modeling and legal theory, there is the need to map the concepts of the two frameworks. The embodiment relationship is used to identify which actors of the system correspond to which subject addressed by the law. To address the need of an actor to comply with a given NP, the concept of realization is introduced, clarifying this link from the intentions of the actor (IE), to the legal entity that needs to be satisfied (NP). A goal doesn't directly realize what the legal text prescribe. The concept of realization is therefore introduced to represent and clarify the distinction between what the actor actually realizes (Realization) and the corresponding action that the law envisages (realizedBy association).

The articulated structure of the law expresses not only rights but also the conditions, exception and other constraints that outline the whole legal document. To represent this variability, the dominance relationship is introduced in the Nòmos metamodel. A relationship characterizing the priority of a right over another, allows the language to represent the conditional clauses typical of legal documents.

2.1.2 Backward forward label propagation algorithms

Backward and forward label propagation algorithms are two important techniques used in GORE to evaluate different aspects of a goal model. These algorithms stems from well established techniques used in Artificial Intelligence (AI) used to train such networks in order to perform some tasks [Nilsson, 1971]. Forward propagation algorithms involve assigning labels to some node in a network/graph, and using the semantics of the relationship in the network to propagate values to other nodes of the network. Backward propagation algorithms search for a consistent assignment of labels to all the nodes of a network. In simpler terms, forward propagation algorithms can be used to perform "what-if" type of analysis, and evaluate the final status of a network given a specific input. Backward propagation algorithms instead can be used to perform "is-this-possible" type of analysis, and evaluate if it exists a configuration/input to the network such that the specific output is generated.

Label propagation algorithms have been used in goal models to analyze and reason over goal models (see Horkoff and Yu [2013] for a detailed review and classifications of the analyses and approaches in goal-modeling).

Sebastiani et al. [2004] propose qualitative algorithms using forward and backward algorithms to analyze combinations of satisfaction values for leaf goals which would result in thee desired values for the high-level goals. A SAT solver is used to identify the satisfiability values of the nodes, and allows for the specification of constraints on goals.

Giorgini et al. [2005b] uses both forward and backward reasoning to incorporate goal model analysis procedures in the Tropos [Bresciani et al., 2004] framework. Forward reasoning is used to evaluate alternatives in the system to-be model used in the late requirements engineering stage of Tropos. Backwards reasoning is used to find the acceptable alternative with the lowest cost.

In the paper [Jureta et al., 2010], the authors uses backward algorithms to find solutions to the requirements engineering problem. The author revise the core problem of requirements engineering in terms of finding a set of tasks and domain assumptions which satisfy all mandatory goals, quality constraints, and ideally also satisfy at least some of the preferred and/or optional goals and quality constraints. Solutions are then compared based on the preferred/optional goals and quality constraints.

Horkoff and Yu [2010] presents a backwards, iterative, interactive evaluation procedure propagating backward from high-level target goals. The proposed procedure in the work formalizes forward and backward propagation rules of the i* [Yu, 1997a] framework in Conjunctive Normal Form (CNF), iteratively applying a SAT solver and human intervention to search for an acceptable solution.

In the same year, Amyot et al. [2010] have proposed a qualitative and quantitative evaluation of the satisfaction levels of the actors and intentional elements in GRL (Goal-oriented Requirements Language) [Amyot, 2003]. This approach relies on three variations of forward and backward propagating algorithm that (quantitative, qualitative and hybrid) have been formalized and provide ways for the modeler to reason about the effectiveness of design alternatives.

2.1.3 Goal-Oriented Requirements Engineering (GORE)

Goal-Oriented Requirements Engineering (GORE) is approach in Requirements Engineer (RE) [van Lamsweerde, 2001], where the needs of stakeholders are represented as goals [Dardenne et al., 1993] — desired state-of-affairs — and are elicited, modeled and analyzed in order to evaluate the requirements that the system-to-be must have. In the last two decades many goal modeling languages have been proposed in the literature. We review here some of the most popular.

The NFR framework [Mylopoulos et al., 1992; in Software Engineering. Conceptual Modeling Foundations and Applications, 2000] is a modeling framework for representing human intentions in technical systems. The framework represents non-functional requirements as 'softgoals', which are goals whose satisfaction criteria is not clear cut. AND- and OR-decompositions, together with the interdependencies between softgoals, are captured in graphs. The impact of decisions is propagated through the graph to determine how well a chosen target system satisfies its NFRs.

The KAOS methodology (Keep All Objectives Satisfied) [van Lamsweerde, 2004, 2001, 2009] introduces a formal goal framework which has been developed and extended over the years. The main concept of KAOS is the goal model: a connected graph of nodes (goal, action, entity, agent, event) where the edges represents the relationships between the entities. Like in similar approaches goals can be and- or-decomposed and are the basis for reasoning about alternative solutions about the system design. Two of the important extensions to KAOS are the obstacle analysis [van Lamsweerde and Letier, 2000] — providing modeling constructs and reasoning techniques to identify and manage pre-conditions for the non-satisfactions of goals — and conflict analysis [van Lamsweerde et al., 1998] — the task of identifying (strong) logical conflicts among goals which would result in an inconsistent system.

The GBRAM technique (goal-based requirements analysis method) [Antón and Potts, 1998] is a method in requirements engineering used to help the analyst uncover hidden issues, goals and requirements. This method uses systematic questions and scenario to refine initial goals, and a set of heuristics is used to identify and analyze goals and constraints. Initial goals can be relaxed by considering obstacles, i.e., something that can happen that could hurt a goal.

The i* (distributed intentionality) framework [Yu, 1996, 1997a] is a modeling language based on the NFR-framework, which incorporates the concept of (hard) goal, resources, and dependencies between actors. The central concept in i* is the actor, an active entity of the system that has strategic goals and performs actions to achieve them. A goal is a strategic interest of an actor, a state of the system that a stakeholder would like to achieve. A softgoal is also a desired state of the system without a clear-cut criteria for whether the condition is achieved. A task specifies a particular course of actions that can be executed in order to satisfy a goal. A resource is a physical or informational entity. The two main components of an i* model are the Strategic Dependency (SD) model and the Strategic Rationale (SR) model. The former is used to describe the dependencies that actors have to accomplish their work. The latter models the relationships within an actor, describing the actors reasoning capabilities and his intentional elements.

Tropos [Bresciani et al., 2004] is an agent oriented system development methodology which uses i* for analyzing requirements. The Tropos methodology makes an important distinction in the requirements analysis process and differentiates two phases: early and late requirements analysis. Early requirements is concerned with understanding the organizational context of the system-to-be. Late requirements analysis is concerned with the definition of functional and non-functional requirements of the system-to-be. These two phases — together with architectural design, detailed design, and implementation — constitute the methodology that guides the analyst from the requirements to actual implementation.

A simplified version of i* was used to create GRL [Amyot, 2003] (Goal-oriented Language), which together with Use Case Maps (UCM) constitute URN (User Requirements Notation). In GRL the main intentional elements of i* are used (goal, softgoal, task, belief and resource) and together with the main five relationships between elements (means-ends, decomposition, contribution, correlation and dependency), it is possible to express conflict between goals and helps to make decisions that resolve conflicts. GRL defines seven contribution types: make, help, some positive, unknown, some negative, break, hurt. Their formalization is used to perform qualitative and quantitative analysis of the models to evaluate the different satisfaction levels of the elements (denied/satisfied, weakly denied/satisfied, conflict, unknown or none) [Amyot et al., 2009].

Techne [Jureta et al., 2010] is a recent requirements modeling language developed to address some of the limitations of the existing languages and analysis technique in goal modeling. The main concepts in Techne are goals, soft-goals, tasks, domain assumptions and quality constraints. These elements can be considered as being mandatory or nice-to-have. Given the three relationships between elements — inference, conflict, and priority — together with decomposition and aggregation of elements, are used in the reasoning algorithms to explore all the possible candidate solutions that satisfy at least the mandatory requirements.

2.1.4 The Business Intelligence Model

The Business Intelligence Model (BIM) [Horkoff et al., 2014, 2012] is a goal-oriented language for modeling organizational strategies proposed to bridge the gap between the data and business domain for BI purposes. The purpose of BIM is to enable decision making during strategic business analysis by using familiar concepts from existing languages (goals, processes, situations, influences, and indicators).

The BIM ontology is founded on four concepts: Situations, Tasks, Indicators and Entities. Situation is a concept adopted from SWOT analysis, and in BIM it is a partial descriptions of world states[1] which may affect Business Objectives. Tasks are used to represent processes or a set of actions. Indicators are used to link business schemas to the data of different data sources. Indicators are linked to Situations in order to evaluate why there is interest in the indicator, and are used to measure tasks. Entities are general concepts used to represent information relevant to the business schema and which can be ad-hoc specialized in the modeled context.

In BIM Situations are specialized in Goals, Operational Situations, and Domain Assumptions. Goals are Situations desired by the modeled organization (e.g., "Increase revenue from marketing"), which may/may-not be pursued and have and evidence value. Like in most goal modeling languages, Goals can be refined into actions that help achieve them: Tasks. Domain Assumptions are Situations that are assumed to hold in the real world.

Four relationships are included in the BIM ontology: Influence, Refines, Evaluates and Measures. The influences relationship is used to represent the (transmission of) (un)favorable effects on Situations. For example the Situation "Economic crisis" has a partial negative influence on "Increase revenue". The refines relationship helps decompose concepts into other: a Goal can be refined into other Goals, Domain Assumptions or Tasks. Except for Situations, refinement is the only way in which Things can accumulate evidence. In BIM refinements are disjunctions by default unless they are specifically AND-marked. Evaluate is used to link Indicators with Situations, and represent that the value of the indicator evaluates evidence for/against the occurrence of the target Situation. For example, the Indicator "Bid-offer spread" evaluates the Goal "Minimize the size of the transaction cost". The measure link is intended to be an abstract relationship that associates a particular Indicator with the Task that it measures. For example the Indicator "Bid-offer spread" can be associated with the data produced by the Task "Obtain market data". In [Horkoff et al., 2012] the official formal semantics of the language is provided.

[1]`http://plato.stanford.edu/archives/fall2008/entries/states-of-affairs/`

2.2 Related work

The goal of this section to give an introduction to the topics related to our work: compliance in requirements engineering, and conceptual modeling.

2.2.1 Requirements engineering and regulatory compliance

The problem of regulatory compliance in requirements engineering is being investigated for several years now. Recently law has been recognized as a cause for the evolution of software requirements that need to be constantly aligned with applicable laws. For example, Ciaghi et al. Ciaghi and Villafiorita [2011] did an extensive analysis of the Italian body of laws, and evaluated how legal production has changed in Italy in the past 60 years. They used adapted software metrics to analyze and evaluated the types of act, their stability (number of modifications or updates), their size, and their growth. Even though their approach was not aimed at providing support to this evolution in requirements engineering, their results provide a clear indication of the growing influence that legal evolution has in maintaining requirements aligned to the law. Very recently Maxwell et al. Maxwell et al. [2012a] have proposed a framework for allowing requirements engineer predict which requirements are most likely to evolve because of the HIPAA Security Rule. The authors suggest a taxonomy classification describing how regulations may change (depending on the reason), and suggest targeted heuristics to predict that a regulation will change.

Many solutions to the problem of regulatory compliance of requirements are based on legal-text approaches and focus on the extraction of legal requirements from the law. Maxwell et al. Maxwell and Anton [2009] explain how to extract software requirements from regulatory texts with production rules. The first activity of their methodology is to obtain a production rule model by translating legal text in Prolog, eliminate duplicates and group conditions. Then they identify parameters of each rule (actors, counterparts, source of the rule) and the preconditions for each rule. After refactoring the produced rules (by grouping cases, grouping common conditions, and by reinforcing implied rules) the final result is a production rule model that can be used by the analyst as a repository of if-then statements to access (legal) domain knowledge. Breaux et al. Breaux et al. [2006] have developed a methodology to extract and prioritize rights and obligations from legal texts. The first step is to identify and restate these rights and obligations using Restricted Natural Language Statements (RNLS). Constraints applying to the right/obligation are then identified and mapped into semantic models. Using RNLS, the initial rights/obligation in the text are restated together with a logical conjunction of the identified constraints. With the proposed methodology it is therefore possible to compare rights/obligations, identify ambiguities, and also identify implied rights/obligations.

On top of these text-base approaches, several compliance frameworks have been proposed in RE. One of the first work in this field was Antón et al. [2002] where the authors describe a case study leading to a proposed privacy goal taxonomy. The use of the GBRAM Method (Goal-Based Requirements Analysis Method) is used in Privacy Policies to identify and classify goals deriving from this text (privacy goals) as either privacy protection or privacy vulnerability goals.

The work by Young et al. Young [2010] focuses on privacy policies and proposes a methodology to extract commitments, privileges and rights from these texts. Their methodology takes a privacy policy as input and produces as output a set of software requirements. As pre-processing step, the text was analyzed three times to generate a set of heuristics that would cover all the body of the law. Their proposal is the composed of 3 steps: first the engineer prepares the policy for analysis and produces a set of statements, then all these statements are classified with the given heuristics, and lastly the classified statements are operationalized into requirements.

Breaux et al. Breaux et al. [2009] have proposed a distributed framework that combines delegation and refinement in a system to capture the decisions that the actors of the system make to comply with standards and regulations. The proposed methodology makes sure that regulatory obligations coming from the law are integrated into functional requirements. They achieve this objective by making sure that obligation are satisfied when they are either delegated to other actors (satisfying the original obligation and creating a new one for the delegatee), or refined into functional requirements.

Siena et al. Siena et al. [2009] adopt a goal-modeling approach to compliance and they introduce a new modeling language called Nòmos that provides a legal extension to the i* framework Yu [1997b]. Normative statements (called Normative Propositions) are categorized into the 8 classes of rights derived from the Hohfeldian classification of rights [Hohfeld, 1917]. These rights are linked to the goals through a realization class: compliance is achieved by defining goals that satisfy ('realize') the Normative Proposition that an actor has.

The work by Ghanavati et al. [Ghanavati et al., 2007, 2009] is similar to the one by Siena et al. but using the Goal-oriented Requirements Language (GRL) instead of i*. Their framework extends GRL with compliance links to manage the law and identify which part of the organization model was not following it. They exploited the GRL quantitative and qualitative analysis to analyze the degree of legal compliance and instances of non-compliance. More recently, Ghanavati et al. [2014a] have proposed a legal expansion of GRL to manage normative concepts within the modeling language. They propose a systematic process to extract requirements from regulations that relies on a conceptual model mapping the elements in the law to those in GRL.

Rifaut et al. [Rifaut and Dubois, 2008] also use a goal-oriented approach to compliance and integrate the i* framework [Yu, 1997b] with the ISO/IEC 15540 standard.[2] Their work proposes a taxonomy for business processes, and maps each concepts into an i* concept. The structure of the i* model is then enhanced by the assessment processes offered by the ISO standard, allowing the analyst to improve the requirements elicitation and analysis.

[2]The ISO/IEC 15540 standard defines means to verify conformity of a reference model.

The work by Tawhid et al. [2012] reports on an interesting approach in using GRL for modeling outcome-based regulations in aviations, in order to help regional inspection activities. In this industry application, the authors describe the extension of GRL with qualitative KPIs indicators in order to model these special outcome-based regulations. A set of questions about the regulations was needed in order to obtain an accurate compliance results, therefore providing an important guide for generating compliance assessment regarding the regulation.

Maxwell et al. [2011] they classified 177 cross-references of the HIPAA Privacy Rule, and identified five sets of conflicting compliance requirements and recommend strategies for resolving these kinds of conflicts. A conflict between legal requirements is identified when they differ or contradict each other, and especially in this work they analyze conflict originating from cross-references among legal documents. Based on the acquired experience, the authors then provide four general strategies for guiding the engineers in resolving the conflicting compliance requirements: comply with the most restricting law, consult a legal expert, obligations supersede legal privileges. In the following work by the authors Maxwell et al. [2012b], they propose an adaptability framework to help analysts predict which areas of a law are most likely to change, and therefore choose to focus on the more stable part of the law.

Massey et al. [Massey et al., 2011, 2013, 2014] perform three experimental evaluations to evaluate the accuracy of compliance decisions of software engineering students, to evaluate the readability of policy documents, and to classify the different type of ambiguities in legal documents. The results from the first work [Massey et al., 2011] show that graduate students with a background in software engineering are not prepared to make legal compliance decisions, and also that legal requirements metrics are useful in supporting analysts when they have to take compliance decisions. The second work [Massey et al., 2013] reports on a large-scale analysis of 2,061 policy documents for the purpose of assessing their readability from analysts, and to determine the accuracy of automated techniques in predicting whether such documents contains special type of privacy requirements identified in previous work. The results suggests that the readability of these document is actually an important issue (especially for analysts), and that automated text-mining techniques can actually support the identification of privacy requirements. In their last work [Massey et al., 2014], a taxonomy of ambiguity (lexical, syntactic, semantic, vagueness, incompleteness, referential) is created by the authors and the used in a study with graduate students. In the study the correctness of the identified type of ambiguities is assessed, together with an overall classification of ambiguities in a paragraph from HIPAA.

Gordon and Breaux [2012] propose an empirically validated framework for reconciling legal requirements coming from multiple jurisdictions. The proposed method guides the analyst in evaluating high and low standards of care across multiple regulations. First the analyst extract the requirements from the documents using a legal requirements specification language. Then the analyst is guided through a pair comparison of requirements to evaluate their (dis)similarity, and lastly the framework construct are applied by the analyst to identify and reconcile conflicts among requirements. In [Gordon and Breaux, 2013], the authors propose a preliminary framework and method to identify relevant legal requirements in order to support the analyst in dealing with changes in both requirements and law. The method to build the legal requirements coverage model — used to evaluate if a law contains requirements applicable to the system at hand — consists of three steps. First the regulation has to be translated in the legal requirements specification language (LRSL), then the analyst has to generate logical expressions from the LRSL-translation of the law. Lastly apply the coverage model which will allow the analyst to make assertions about these logical expressions and obtaining a truth value for these expressions asserting some properties interesting to the analyst (like am I a covered entity? am I using sensitive data? and similar questions).

2.2.2 Conceptual modeling and law

The problem of regulatory compliance is a complex issue affecting many information systems, and in these cases abstraction is a common technique that can be used to manage such complexity. Conceptual modeling plays therefore a relevant role in dealing with this problem. The aim of Conceptual modeling is to provide representations of legal provisions with the purpose of allowing full but unambiguous and human-tractable communication of their semantics, in order to support decision making processes about law compliance.

Conceptual modeling is a research area often intersected with another domains. Abstraction is indeed is a common technique that can be used to manage the complexity of problems, and conceptual modeling plays therefore a relevant role in dealing with complex issues like regulatory compliance. Some approaches to the compliance problem use heavy-formalizations [Bench-Capon et al., 2012; Governatori, 2005; Antoniou et al., 1999; Governatori and Rotolo, 2009; Dignum, 1999], while others choose instead to abstract the problem and use conceptual models to manage the complexity of the task at hand. Legal Ontologies are a special

For example, Kiyavitskaya et al. [Kiyavitskaya et al., 2008] propose an extension to Cerno methodology with a tool-supported process to transform the semantic model into a set of functional and non-functional requirements. They adopted and extended the Cerno framework to help the engineers identify the basic concepts and perform a semantic annotation of legal text using the methodology of Breaux et al. [Breaux et al., 2006]. van Engers and Nijssen [2014] have extended the four basic hohfeldian categories with temporal relationships and explicit events. This extension to the semantic conceptual model for Hohfeld allows them to analyze regulations using a state-transactional view, and enables a service-oriented perspective where regulatory sources are compared with the life events of a service to check for its compliance.

In requirements engineering, the work by Ghanavati et al. [Ghanavati et al., 2014a] provides a meta-model for the proposed Legal GRL modeling language, and describe a conceptual mapping between the meta-model of their language and the hohfeldian's. Siena et al. [2009] also shows how the use of a dedicated conceptual model for supporting regulatory compliance in i* models, is useful in achieving a compliant design for a software. Tawhid et al. [2012] propose an extension of the meta-model of the GRL language with KPIs indicators in order to model outcome-based regulations.

In the field of Business Process and Enterprise Modeling for example, Kharbili et al. [2011] propose CoReL, a domain specific language for the domain of compliance decision-making. Their language requires first that legal expert interpret the regulation and formulate — according to the context at hand — a set of guidelines called Compliance Requirements. These requirements are then incorporated in their methodology and translated into rules that constrain the different modules of the policy. Butler and McGovern [2012] propose a process-base conceptual model on the design and adoption of GRC (governance, compliance, and risk) in Enterprise Systems. Shamsaei et al. [2010]; Shamsaei [2011] presents an extension of GRL with indicators to model regulations and organizational objective. Their proposed method uses URN to measure compliance and integrates it with KPI (key performance indicators). The innovative use of KPI for measuring compliance of business process, is used in conjunction with goal modeling (GRL) for modeling organizational objective and evaluate when business processes violate some norms. In the work of Schultz [Schultz, 2013] it is claimed that most approaches dealing with compliance, take quite a technical perspective and a comprehensive conceptualization of compliance for auditors is missing. They propose a conceptual model based on interviews with experts and online surveys, which therefore takes in consideration the stakeholders' perspective of compliance in business process. Knuplesch et al. [2010] show how abstraction can improve the efficiency of compliance checking algorithms. The authors present an approach for dealing with the problem of state explosions when data-related compliance rules are checked with model checkers. By abstracting the large number of states the data can be, the authors show how to manage the problem of compliance of business process.

Legal ontologies

The complexity of legal knowledge increases as new technologies relaying on that information are proposed to manage and help the interdisciplinarity that this field offers. Domain specific ontologies are therefore crucial to develop accurate and coherent tools for handling this legal knowledge. In her book, Núria Casellas Casellas [2011] has recently provided a comprehensive overview of legal ontologies in the current literature, and she focuses on legal ontologies as the form of representation and formalization of legal knowledge.

For example Rodríguez-Doncel et al. [2014] presents an ontology-driven approach in aviation and presents an interesting application to support the Air Transport Passenger's Incidents and Rights. García and Gil [2009] presents a copyright ontology in order to better support machine-understandable licences. Asaro et al. [2003] have developed a domain ontology for the Italian Criminal Law to support the judge's activity. Gangemi et al. [2003] have proposed a Core Legal Ontology based on the extended DOLCE foundational ontology to support the definition of domain ontologies or legal support systems. Hoekstra et al. [2007] have proposed a legal core ontology (LKIF - Legal Knowledge Interchange Format) developed int he context of the Estrella project. Rubino et al. [2006] have proposed an OWL ontology of fundamental legal concepts that includes and defines a thorough set of deontic modalities.

Having identified the laws governing an IS domain, Khadraoui et al. [2009] proposes a method for extracting an ontology representing the organizational roles, which are the legal subjects of the laws governing the IS domain. For each role, a model of the domain artifacts and processes under its responsibility is defined. This IS ontology model provides a reference for the designers to discuss and understand the ways in which they view and interpret the institutional domain. Valente et al. [1999] presents an ontology-based architecture for managing and reasoning with legal information. Part of the legal knowledge consists of the description of norms which are expressed as normative functions. Hage and Verheij [1999] presents a top-level ontology of the law based on the characteristics that the law is a dynamic interconnected system of states of affairs.

2.3 Related research areas

2.3.1 Normative multi-agents systems

The research area of normative multi-agent systems can be defined as the intersection of normative systems and multi-agent systems Boella and Torre [2006].

The perspective taken in this field is that norms act as behavioral constraints that regulate and structure social order within a multi-agent system. Typical problems in this field are therefore to define and derive those rules and monitor the agent's behaviour as complying to these rules. For example, the formalism proposed by Governatori et al. Governatori and Rotolo [2009] allows to check whether agents' behaviour complies with the rules regulating them. Their work models agents' behaviour in terms of processes, and extends the proposed Process Compliance Language with structural annotations describing deontic obligations to be met, or the effects to be achieved. Compliance to the rules can be verified through the deontic component of the proposed language. Dignum Dignum [1999] also exploits deontic logic for modeling norms, their violations, and possible reactions to these violations. In this work the social behaviour of an agent is divided in three layers (private, social, and contract), and the use of deontic logic for modeling concepts in these layers gives the opportunity to specify explicitly what should happen in cases of violation of the obligations. The architecture therefore pursues agents' compliance as the consequences of violating a convention lead to a state that is less preferred than the state that is reached by adhering to the convention. The approach used by Singh Singh [1999] is instead to model normative concepts (like conventions, obligations, pledges) as different kinds of social commitments. This core concept of social commitment is indeed used to characterized 'traditional' muti-agents systems, but also to model normative concepts and the Hohfelidans relationships between agents.

Recently, Boealla et al. [Boella et al., 2011] have proposed Eunomos, a legal document and knowledge management system based on ontologies and legislative XML. The software is used to crawl daily the web to collect and classify new regulations. The newly identified regulation is first translated in legislative XML, then cross-references are extracted in order to build a link between the extracted legal knowledge and its legislative source. A dedicated legal ontology tool is used to extract and model the legal concepts in the regulation. The aim of the project is to enable users to identify regulations coming from different sources, and find the right definitions and explanations of legal concepts in a given context.

Antonini et al. [2013] discuss the challenges of norms in medical law and clinical guidelines. They propose an extension of the Eunomos framework [Boella et al., 2011], for managing the interplay between law and specialist domains. Their analysis points out three main issue: first an ontological difference of descriptions, then a different status of norm (absolute, defeasible, advice), and lastly a different type of normative reasoning that is needed at the different levels (goals vs procedures). Their proposal to use the Eunomos system for reasoning on actual cases, shows the need for representing the different functional roles in the analyzed domain.

Singh [2013] presents an approach for governing socio-techincal systems that uses the concept of norms for their regulation. Their formal model considers five type of norms (commitment, authorization, prohibition, sanction and power), which are constituted from an antecedent and a consequent. Based on the value of these two parts, the state of a norm is deducted. This set of normative clauses supports different type of analysis of organizations and norms.

2.3.2 AI&Law

The research in the field of AI&Law is mainly concerned with problems related to the logical formalization and representation of elements of the law and legal reasoning. The formalization of legal reasoning and legal argumentation is being investigated in this field for many yeas, and many solutions have been proposed so far.[3] For example, one of the first solution to formally define a model of legal argumentation was proposed by Gordon et al. in 1997 Gordon and Karacapilidis [1997]. The Zeno argumentation framework structured information relevant to a discussion using a dialectical graph. Through this structure elements can be evaluated and it is possible to evaluate the winning and losing position in a debate.

Many solutions propose formal models of the law based on deontic logic (see Bench-Capon et al. [2012]; Governatori and Rotolo [2009]; Dignum [1999]). Deontic logics are intended to represent obligations, prohibitions, and permissions McCarty [1989], all considered central concepts in law. Although this basic ontology of deontic logics seems well understood, this remains a active area of research, in particular into more expressive deontic logics and efficient reasoning proce-

[3]This consolidated field has recently published a retrospective paper summarizing the most relevant works in this community over the past 25 years Bench-Capon et al. [2012].

dures Hansen et al. [2007]. For example Governatori Governatori [2005] shows how contracts' clauses describe the obligations and other normative positions for the parties involved in a contract as well as compensatory clauses. In this work the author shows how the norm is analyzed and represented using a logical framework based on deontic and defeasible logic. Deontic Logic is used to define modalities, roles and behaviours of the contract, while defeasible logic helps the clarification of inconsistencies. The translation of these logical rules into machine readable format, allows the automatic monitoring of violations in the contract. Another logic used in the formal representation of law is defeasible logic. Defeasible Logic instead provides five different kinds of knowledge representation constructs: facts, strict rules, defeasible rules, defeaters, and a priority relationship to indicate what conclusions to draw in case of contradiction. Defeasible logic has been applied to the modelling of regulations, business rules and contracts Antoniou et al. [1999]; Boella et al. [2009], but computational complexity seems to limit its adoption Johnston and Governatori [2003].

2.3.3 Compliance of business process and enterprise modeling

In the fields of Business Process, many works focus on the evaluation of their compliance with norms (for a review on the state of the art in this field, see for example Fellmann and Zasada [2014] or Becker et al. [2012]). For example, Sadiq et al. [2007] is an example of compliance-aware design of business processes. Their approach is built on a requirements modeling framework that propagates these requirements onto business processes. Policies are meant to be directly extracted from regulations, though a translation from legal experts is required beforehand. This translation introduces a layer between the requirements model and the enforcement of compliance policies. Such layer would allow for example to exchange policies or discover policy conflicts between business processes. Liu et al. [2007] presents an approach based on temporal logic together with pi-calculus for the evaluation of compliance of business processes. Business Models are specified in BPEL (Business Process Execution Language) and using pi-calculus they are translated in a FSM representation. The Compliance rules are instead formalized in BPSL (Business Property Specification Language) and then translated in linear temporal logic. Model checking techniques are then used to verify whether the business processes comply with the imposed regulations. Awad et al. [2008] also describe an approach for automated compliance checking of business processes based on temporal logic. The approach focus on evaluating compliance with respect to the ordering constraints of activities. In this context, compliance rules are expressed as queries that determine the set of process models that needs to be further evaluated for compliance. Witt et al. [2014] propose a tool to support a requirements validation and verification framework for business process compliance. Their approach allows to graphically specify formal requirements on the basis of business process model. The tool uses Graphical Computational Tree Logic (G-CTL) to model the validation rules regarding temporal behaviour. Rules can be defined in detail or general manner, allowing their reuse across different models.

2.3.4 Security and compliance with privacy laws/standards

The problem of aligning a software with regulations initiated because of privacy laws. The problem of privacy has been included and investigated in software security, (e.g., *IEEE Journal on Security & Privacy*, or the *IEEE Symposium on Security and Privacy*). Many proposals in this area focus on cryptography as a mean for data not to be compromised, or standards for security management to be complied with. We present in this section some works in the area of Security and Privacy that are closer to the field of requirements engineering.

Barth et al. [2006] presents a formal framework for expressing privacy expectations and privacy practices. They use temporal logic to define two kinds of norms (called positive and negative) which generalize the concept of permission and prohibition of deontic. These norms are used to allow or deny agents communications if the temporal condition is satisfied. This privacy-oriented framework uses this concept of norm to focus on who the information is about, how it is transmitted, and temporal actions on the information performed at different time and by different users.

Compagna et al. [Compagna et al., 2008] propose to use design patterns to integrate legal and IT concerns. Their methodology proposes to have legal experts describe these patterns and validation processes in natural language. This description is parsed on the basis of a semantic framework (e.g., Breaux Breaux et al. [2006]) and the annotate description is then validated and translated in formal specification. The patterns are revised and validated in order to be used to verify inconsistencies in a graphical model.

Asnar et al. Asnar and Massacci [2011] present a comprehensive method called SI*-GRC, used to analyze and design security controls for organizational settings. Their proposal includes a modeling framework that extends SI* with relationships for security and GRG concerns (permission, trust, risk and treatment), and a tool-supported analysis process for helping the analyst during their proposed analysis method.

Antón and Earp [2004] propose a taxonomy of privacy goal and report on the analysis of 23 Internet privacy policies for companies in three health care industries. The 12 categories in the taxonomy can be described either in terms of Privacy-Protection (desired protection of user privacy rights) or Privacy-Vulnerability (potential for invasions of privacy). The identified taxonomy is used for comparing and analyzing privacy-policy statements, and discover inner internal conflicts in order to ensure their consistency.

Banescu et al. [2012] proposes a new metric for evaluating compliance, based on the number of deviations from the specifications of business processes. The authors define four patterns for identifying different type of deviations that are used to categorize the different type of infringements. A privacy compliance technique uses this patterns to first identify mishap, and then to measure and quantify their severity. To perform this last task they use a metric (based on accessed data, executed action, user role) that evaluates the 'privacy' distance between the intended and actual behavior.

Massacci et al. [2005] presents a case study of the Secure Tropos methodology Giorgini et al. [2004] to evaluate the alignment between the Italian Data Protection Law and the Data Protection of the University of Trento. Their study suggested that the primitives of Secure Tropos were adequate, and also pointed out how the University Standard often defined which objectives and responsibility the entities should have without identifying who should provide such services. Their analysis suggested that the problem is actually caused by the lack of guidelines/standards considering the functional goals of the organization.

Veeningen et al. [2011] proposes a formal method for analyzing privacy of communications protocols for identity management systems, a recent technology for managing personal data in distributed systems. Their formal framework is proposed to analyze which privacy properties are satisfied by the system. A three-layer model is used to represent personal information (content, information, and object layer): personal data represented in these layers are used to identify identity-related properties of an actor. This properties are then formalized and integrated in each subsequent layer, and using cryptographic primitives, messages of the protocol are constructed.

Sen et al. [2014] propose a workflow for privacy compliance in big data systems. Their proposal includes a language for helping policy authors and privacy policy champions specify privacy policies, as well as a data-inventory for big data system. This system maps every dynamic schema element in the language, therefore allowing the workflow to automatically check for compliance of the data schema by imposing restrictions on how the data/information should be handled.

Chapter 3

Language

3.1 Introduction

Nòmos 3 is a formal language designed to assist the modeller in the following activities:

- the representation of law and requirements, using a small set of primitive concepts and relationships,
- the representation of alternative ways to comply with the law,
- the representation of the roles who have to comply with the law.
- automatically finding which law applies to which situations that the requirements of the system-to-be produces,
- automatically finding which situations do comply with applicable law, and which fail to comply,
- automatically finding which roles have to comply with applicable law, and how they influence the compliance with the law.

We say that the language is lightweight because it is based on a small set of concepts and relationships, and is propositional. Nòmos 3 models are made of typed logical variables, each referring to a statement in unrestricted natural language, and a predefined set of relationships to indicate interactions between the statements.

The basic concepts of Nòmos are are *situations*, *roles*, and the relationships characterizing their interaction with the *norms* representing the law. A situation is a proposition [Stanford Encyclopedia, 2005], that is, a partial description of a world. Two relationships — satisfy and break — are used to say how bringing about a situation influences other situations. By asserting that some situations are satisfied and others failed, the satisfy and break relationships make it possible to compute if other situations are satisfied or failed. In short, the basic building blocks in Nòmos 3 models represent propositions, and indicate how the satisfaction of some propositions influences that of others.

Roles are used to represent the addressee of norms, as well as the roles in the requirements of the system-to-be. Relationships are used to link the role with the norm addressing it, and the desired requirements of the system-to-be. These relationships are used to characterize the responsibilities that the roles have — represented in terms of situations assigned to the role — and to evaluate what the roles have to do in order to comply with the law and fulfill the objectives in the system-to-be.

Nòmos 3 models represent laws by representing its most atomic pieces, called *norms* Wikipedia [2005] (duties or right). We use situations to represent the conditions that make a norm applicable, and also the conditions that correspond to doing what the norm ask to do (which we refer as 'satisfying' the norm). For example if the situation "Car is on highway" is satisfied, then the duty to "Turn headlights on" is applicable. Satisfying such duty, calls for the satisfaction of a situation like "The car's headlights are turned on". Compliance therefore correspond to satisfying an applicable norm.

In Nòmos 3 the requirements of the system-to-be are represented in terms of goals that the actors in the system want to achieve. By representing the goals in terms of situations — and by modeling the relationship characterizing how a set of situations influences the satisfaction of the goals — it is possible to relate the goals with the norm they make applicable or satisfy.

3.2 Concepts and relationships

A Nòmos 3 model is conceptually comprised by two parts: a *legal model* and a *requirements model*.

The legal model represents a law in terms of: (i) the norms, atomic fragments of law, it includes; (ii) the conditions that make norms complied with or violated; and (iii) the legal roles responsible for complying. The legal model relies on the concepts of Norm, Situation, and Legal Role.

The requirements model represents the requirements that are going to be analyzed with respect to a piece of law. Nòmos 3 adopts modeling constructs typical from a goal-oriented requirements engineering (GORE) approach van Lamsweerde [2001], and represents the requirements model in terms of (i) the goals that some social roles (i.e., the stakeholders of the system-to-be) want; (ii) the social roles who want to achieve such goals; (ii) the conditions holding for the goal to be achieved, or knowing to hold in the domain. The requirements model relies on the concepts of Goal, Situation, Domain Assumption, and Social Role.

In the following we present the concepts and relationships between them that respectively capture the semantics of a fragment of law and represent the requirements. The running example describes the scenario of a software for online sellers in the U.K. (for the requirements model) who need a software to manage their online-shop while being compliance with norms regarding VAT and invoices (for the legal model).

3.2.1 Concepts

Situation. Situations are states of the world represented by propositions, like “It’s Christmas season” or “John likes Mary”. For example, when the Situation s_1 “Customer is charged for item price” holds, propositions such as “Good is sold” are true, while “Good is donated” is false, and nothing can be said about the proposition “Paolo likes Marta”.

Situation is a primitive concept in our language which is used to represent the antecedent and consequent of a Norm, and the responsibilities assigned to Roles (e.g., Goals). Situations function as glue between the two models — they represent conditions holding in both model — and allow to reason about which Situations should hold in the models in order to comply with some Norm while making some Goals achieved.

Norm. A **Norm** is an atomic fragment of law with deontic status, e.g. “VAT-registered business must issue valid invoices when selling good”. A Norm is composed of five elements: *type*, *holder*, *counterpart*, *antecedent*, *consequent*: *type* identifies the deontic status of a Norm as a right or duty (a duty in the example above). *holder* is the Legal Roles addressed by the Norm (e.g., the `VAT-registered business`) and who has to satisfy the Norm if the Norm applies. *count* is the optional Legal Role whose interests are helped if the Norm is satisfied. *antecedent* and *consequent* of a Norm are the conditions (Situations) that represents what trigger the Norm to be applicable (“Good is sold”) and satisfied (“Valid invoice is issued”).

- A Norm’s *type* allows us to specialize the Norm concept according to a legal ontology of choice. We assume that at least two specializations of a Norm are provided by the ontology — duty and right — and a type is always be specified. A duty entails that if the Norm is applicable (the antecedent of the Norm holds), then the holder must bring about the consequent. The VAT-registered business must bring about the consequent of the Norm (“Valid invoice is issued”), when the antecedent holds (“Good is sold”). A right states that if the Norm is applicable (“Valid invoice is issued”), then the holder can choose to bring about the consequent of the right (but the norm is not violated if it does not happen). Rights are used to exclude the need to comply with a duty.The ultimate purpose of our proposed language is to evaluate compliance. In natural language, we say that duties are complied with, whereas rights can be exercised. In this paper, we use the phrase comply with a Norm to mean both “comply with a duty”, if the Norm is a duty, and “exercise a right”, if the Norm is a right.

- The Norm’s holder specifies the Legal Role in charge of complying with the Norm. In the example above the `VAT-registered business` is the Legal Role that the Norm says has to comply with the duty. Despite the challenges and peculiarities of legal text — which can be ambiguous or omit some details — in our conceptual model we assume that it is always possible to identify the holder of a Norm. In the current work, we focus only on Norm with a single holder, and not on group or multiple-holders Norms.

- The Norm's counterpart specifies the *optional* Legal Role who benefits if the Norm is satisfied. For example in the example above, the Legal Role of `VAT-registered client` might be a counterpart. Given the challenges and peculiarities of legal text where this element is often not specified, in our conceptual model we do not assume that the Counterpart is always present.

- The Norm's antecedent describes the conditions under which the Norm takes effect/is triggered: if the Norm takes effect and the holder can/must bring about the consequent, we say the Norm is applicable. For example, the Norm above of the British legislation states that "VAT-registered businesses must issue correct VAT invoices for their goods or service that is sold". In this Norm,"Good or service is sold" is the antecedent and, when it holds, the prescription of issuing a VAT invoice holds (i.e., it must be satisfied). If the Norm has a antecedent, but the antecedent does not hold, then the Norm is not applicable. In the example, if no good or service is sold, there is no need to issue a VAT invoice. We assume that if the Norm has no specified antecedent, it is always applicable.

- The Norm's consequent represents the conditions under which the Norm is respected or, in other word, what the Norm wants the holder to do/to be able to do: we call *satisfaction* and we say that a norm is *satisfied* when these conditions hold. In the example above, "issuing a VAT invoice" is the Norm's consequent. Satisfying a Norm is different from complying with it. If the VAT-registered business issues an invoice but no good is sold, there is no compliance for that norm.

Role. We use Roles to represent the distribution of responsibilities in the legal and requirements model. Responsibilities are represented as Situations assigned to a Role, and the satisfaction of such Situations determines the fulfillment of the Role. A Role is a primitive concept and we distinguish two types of Roles.

- **Legal Role** is a Role in the legal model who is addressed by a Norm. In the example above, the `VAT-registered business` is a Role that the law addresses as being the holder of the duty. Instances of Legal Role appear as the holder and counterpart as parts of instances of the Norm concept. We assume that if a Legal Role exists, there is at least one Norm addressing that Legal Role. The Situations that satisfy an applicable Norm are assigned to the Legal Role holder of the Norm. Equivalently, we then say that the Legal Role is responsible for the Situations satisfying the Norm. For example, when the duty to issue

VAT invoices applies, the Legal Role of ‘`VAT-registered business` is responsible for the Situation satisfying the Norm “Valid invoice is issued”.

- **Social Role** is a Role in the requirements model who wants to achieve some objectives (called Goals). This concept is borrowed from the literature on goal-oriented requirements engineering and it represents the specialization ‘Role’ of the concept of Actor in i*: an “abstract characterization of the behavior of a social actor within some specialized context or domain of endeavor”.[1] In the example above, the “online-seller” is the Role in the requirements model who has some objectives he would like to achieve. The Social Role has two responsibilities.

  First some social responsibilities: a Social Role is responsible for the Situations that satisfy the Goal he/she wants. For example the online-seller is responsible for bringing about the situation “Package is shipped” satisfying his objective to “manage shipping”. We assume that if a Social Role exists in our model, he/she desires at least one Goal.

  Secondly the Social Role has some legal responsibilities: in our context Social Roles are linked to Legal Roles — we say Social Role coplays a Legal Role — to represent the fact that Social Roles in the requirements model have to comply with the Norms. Since legal text and Norms address Legal Role, the link between the two Roles is what makes a Social Role have legal responsibilities. For example, the Social Role of `Online seller` it is also responsible for the Situations assigned to the Legal Role of `VAT-registered business` it co-plays. Similarly the Social Role of ‘`Online customer` could coplay the Legal Role of `VAT-registered client`, if there is a norm addressing such special Legal Role and its rights/duties. We assume that a Social Role does not have to coplay a Legal Role — it is possible that for example, the Social Role of `System Admin` has no legal responsibilities — and a Social Role can coplay more than one Legal Role (acquiring the legal responsibilities of both roles). Similarly, a Legal Role does not have to be coplayed by a Social Role — it is possible that no Social Role in the requirements coplays the Legal Role of `VAT-registered business with a second-hand margin scheme` — and a Legal Role can be coplayed by more than one Social Role.

Goal. A Goal represents a requirement that a stakeholder (Social Role) wants to achieve. This concept is borrowed from the literature on goal-oriented requirements engineering and it represents “an intentional desire of an actor”.[2] For example, one of the Goals of the `Online seller` is “Manage sold items”. In our model we assume that each Goal is desired by at least one Social Role.

[1] http://istar.rwth-aachen.de/tiki-index.php?page=Role&structure=i*+Guide, accessed in May 2015.
[2] http://istar.rwth-aachen.de/tiki-index.php?page=Goals+(Hard+Goals)&structure=i*+Guide.

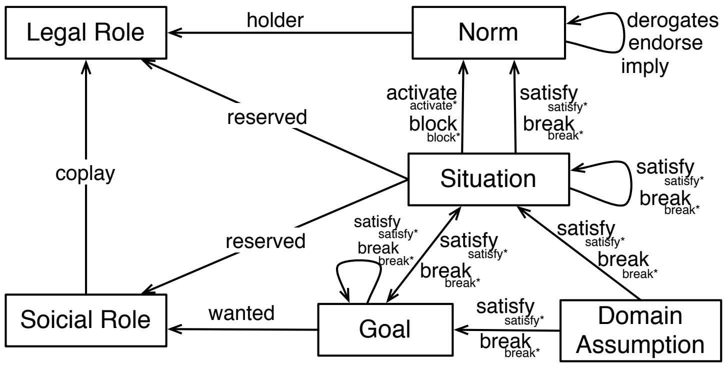


Figure 3.1: Relationships of the Nòmos 3 language.

Domain Assumption. A Domain Assumption (DA) represents a general state-of-affairs that is assumed to hold in the domain for a solution to a goal to work. In the scenario of the system to help British online sellers manage selling and VAT, examples of domain assumptions are "Customers have a postal address for shipping", or "The address of the Online Seller is located in the U.K.".

3.2.2 Relationships

Between Situations.

satisfy is a relationship between two Situations such that when the source Situation holds, the target Situation also holds. For example this relationship can be used to relate source Situation "Shipping address is in Italy" to target Situation "Shipping address is in EU".
The satisfy* relationship states that when the source Situation does not hold, the target Situation holds. If in our domain we only know that extra-eu shipping is tracked, we can use the satisfy* relationship to specify that when "Shipping address is in EU" does not hold, then the Situation "Shipping is tracked" holds.

break is a relationship between two Situations such that when the source Situation holds, the target Situation does not hold. This relationship can be used to relate source Situation "Shipping address is Hong-Kong" to target Situation "Shipping address is in EU".
The break* relationship is defined analogously to satisfy*.

Between Situations and Norms.

activate is a relationship between a Situations and a Norm such that when the source Situation holds, the Norm's antecedent holds and we say the Norm is applicable. activate* is defined analogously to other *-ed relationships. For example, when the Situation "Good is sold" holds, then the Norm "Duty to issue a valid invoice" is applicable (the antecedent of the Norm is satisfied.)

block is a relationship between a Situations and a Norm such that when the source Situation holds, the Norm's antecedent does not hold. Similarly, the relationship block* represents that when the source Situation does not hold, the Norm's antecedent does not hold. For example, when the Situation "Good is offered for free", then the same Norm does not apply (the antecedent of the Norm is not satisfied)

satisfy is a relationship between a Situations and a Norm such that when the source Situation holds, the Norm's consequent holds and we say the Norm is satisfied. Similarly, the relationship satisfy* represents that when the source Situation does not hold, the Norm's consequent holds. In the example, the Situation "Valid invoice is issued" is what makes the Norm's consequent satisfied and we say the Norm is satisfied.

break is a relationship between a Situations and a Norm such that when the source Situation holds, the Norm's consequent does not hold. Similarly, the relationship break* represents that when the source Situation does not hold, the Norm's consequent does not hold. In the example, the Situation "Purchase receipt is issued" does not make the Norm's consequent satisfied.

Between Norms.

endorse is a relationship between Norms such that when the source Norm is compliant, the target Norm is applicable. For example, all VAT-registered business have the duty to keep all purchase invoices for items they buy for the company. So when the duty is compliant (and all invoices of items bought for the company are kept), then it is applicable the right for `VAT-registered business` to reclaim VAT.

imply is a relationship between Norms such that when the source Norm is compliant, the target Norm is also compliant. For example, if the duty to issue a valid invoice is complied with (and therefore a valid issue is issued), then the duty to include the business VAT number in the invoice is also complied with.

derogate is a relationship between Norms such that when the source Norm is compliant, the target Norm is not applicable. For example, when the duty to issue a valid invoice is compliant (and therefore a valid invoice is issued), then `VAT-registered business` to change the price for an item is not applicable.

Between Roles.

coplay is a relationship between a Social and a Legal Role that represents that the Social Role is responsible for the Situations assigned to the the Legal Role. For example when the Social Role `Online seller` coplays the Legal Role of `VAT-registered business`, then it is also responsible for the Situations assigned to that Legal Role.

ocial delegation) is a ternary relationship among two Social Roles (the delegator and the delegatee) and a Goal (the delegatum). We borrow this relationship from Giorgini et al. [2005a] and we include it in our language to support the delegation of (social) responsibility from one Social Role to another. When a Social Role delegates a Goal to another one, the delegatee becomes then the one who wants to achieve the Goal. In other words, once a delegation is in place, the only way for the delegator to achieve its goal is to either have the delegatee achieve it, or do it himself. Cases where the delegatee is not be entitled to bring about the Situation are identified by the reserve schema (see section 5.1.3).

Between Roles and Norms/Goals/Situations.

holder is a relationship between a Norm and a Legal Role such that the Legal Role is responsible for the Situations in the consequent of that Norm. For example the Legal Role of `VAT-registered business` is responsible for the Norm about issuing invoices, so we say that the norm has *holder* this Legal Role.

wants is a relationship between a Goal and a Social Role such that the Social Role is responsible for the Situations satisfying the Goal. For example, one of the Goal of the `Online seller` is to 'Sell items online'. We say the goal is *wanted* by the Social Role who is therefore responsible for the Situations satisfying the goal.

reserved is a relationship between a a Situation and a Role (Legal or Social) such that only the specified Role is entitled to bring about the Situation. For example, only `VAT-registered business` can reclaim VAT.

Between Goals, Situations and DAs. These are special cases of the satisfy and break relationships between Situations: Goals and DAs are Situations, so the relationships between these concepts are in the end relationships between Situations.

satisfy is a relationship between Situations Goals and DAs, such that when the source holds, the target also holds. Similarly, the relationship satisfy* represents that when the source does not hold, the target holds.

break is a relationship between Situations Goals and DAs, such that when the source holds, the target does not hold. The break* relationship represents that when the source does not hold, also the target does not hold.

3.2.3 Visual notation

In figure 3.2 we present the visual notation of Nòmos 3 with an example: the legal model shows an excerpt of the British legislation on VAT record keeping and VAT invoices;[3] the requirements

[3] `https://www.gov.uk/vat-record-keeping/vat-invoices.`

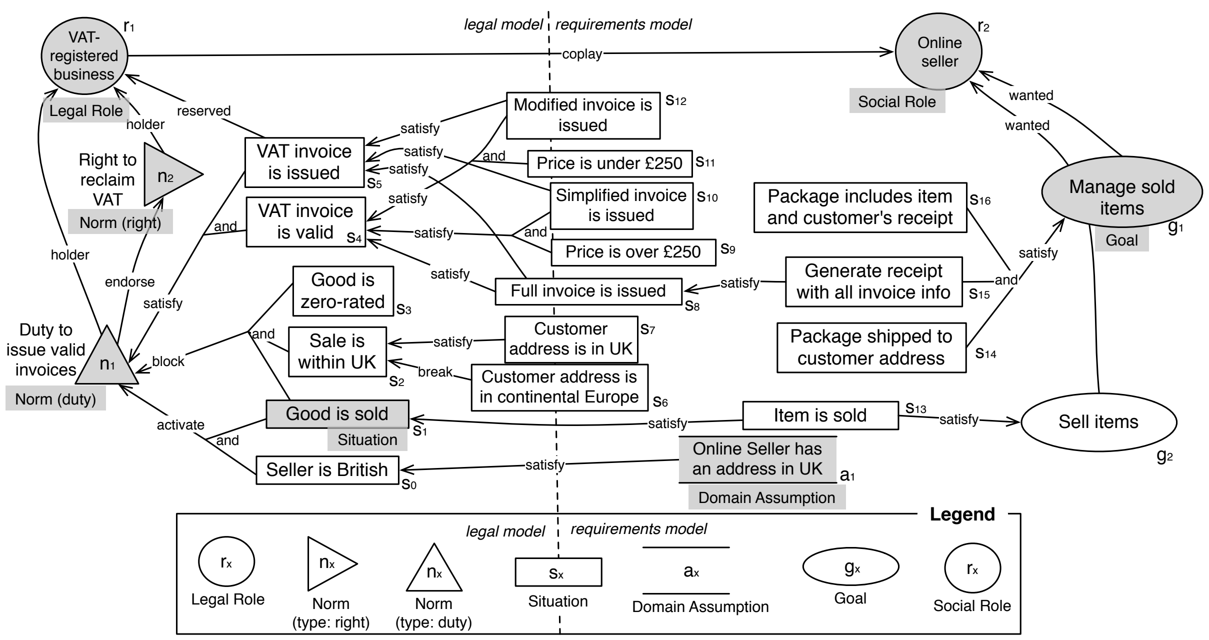


Figure 3.2: Visual notation for the modelling language.

model shows an excerpt of a goal model for a software to help online sellers keep track of goods and transactions.

Norms are represented as triangles — ▷ represents norms of type rights, and △ represents duties — with an indexed label (n_1 and n_1 respectively). Situations are represented as rectangles ▭ containing the proposition associated to the situation, together with an indexed label s_i. Roles are represented as circles ◯ and have an indexed label r_i. Goals are represented as ovals ⬭ containing the proposition associated to the goal, together with an indexed label g_i. Domain Assumptions are represented as open-sided rectangles ═ containing the proposition associated to the situation that is known to hold in the domain, and they have an indexed label a_i. Relationships are arrows where the label characterize the name of the relation. Conjunction and disjunction (resp. means-end and decomposition for goals) are represented explicitly with the tag and/or at the line intersection.

Figure 3.2 can be read as follows.

The legal model on the left shows one Legal Role (r_1, `VAT-registered business`) and two Norms: n_1 is the duty to issue valid invoices, and n_2 is the right to reclaim VAT. Only when the first one is complied with — and valid issues have been issued — then the right to reclaim VAT is applicable: $n_1 \xrightarrow{\text{endorse}} n_2$. The Legal Role addressed by the Norm is the `VAT-registered business` who is then the holder of both Norms: $n_1 \xrightarrow{\text{holder}} r_1$ and $n_2 \xrightarrow{\text{holder}} r_1$. Since the law says that "*only VAT-registered businesses can issue VAT invoices*", the model shows that the Situation "VAT invoice is issued" is only responsibility of that Legal Role: $s_5 \xrightarrow{\text{reserved}} r_1$.

The duty n_1 specifies that *VAT-registered businesses must issue and keep valid invoices for* *sold good*, so when the Situation s_1 “Good is sold” and s_0 “Seller is British”, the duty is applicable: $(s_0 \wedge s_1) \xrightarrow{\text{activate}} n_1$. The duty is satisfied when the invoice is issued and is valid: $(s_4 \wedge s_5) \xrightarrow{\text{satisfy}} n_1$.

A VAT invoice is issued when either the modified- (s_{12}) or simplified- (s_{10}) or full-invoice is issued (s_8): $s_{12} \xrightarrow{\text{satisfy}} s_5$, $s_{10} \xrightarrow{\text{satisfy}} s_5$, $s_8 \xrightarrow{\text{satisfy}} s_5$. Since multiple relationship targeting the same element are treated as being in disjunction, all these relationships are alternative ways for satisfying the target Situation s_5. Similarly, the VAT invoice is valid either (a) when the full invoice is used ($s_8 \xrightarrow{\text{satisfy}} s_4$), or (b) when the simplified invoice is used and the price is over £250 ($(s_9 \wedge s_{10}) \xrightarrow{\text{satisfy}} s_4$), or when the price is below £250 and the modified invoice is used ($(s_{11} \wedge s_{12}) \xrightarrow{\text{satisfy}} s_4$).

Since *You [VAT-registered business] don’t need to issue a VAT invoice if your invoice is only for exempt or zero-rated sales within the UK*, we know that when good is sold (s_1) within the UK (s_2) and it is zero-rated (s_3), the duty to issue the invoice is no longer applicable: $(s_1 \wedge s_2 \wedge s_3) \xrightarrow{\text{block}} n_1$. A sale is within the UK when the address of the customer is in U.K. ($s_7 \xrightarrow{\text{satisfy}} s_2$), and it is not when the customer address is in continental Europe ($s_6 \xrightarrow{\text{satisfy}} s_2$).

The requirements model on the right shows one Social Role (r_2, `Online seller`) and two Goals: g_1 is the Goal to manage sold items, and g_2 is the Goal to sell the items. Both Goals are objectives desired by the Social Role of `Online seller`: $g_1 \xrightarrow{\text{wanted}} r_2$ and $g_2 \xrightarrow{\text{wanted}} r_2$. The DA a_1 states that the `Online seller` has an address in the U.K., and when the DA holds, then it also holds the Situation “Seller is British”: $a_1 \xrightarrow{\text{satisfy}} s_0$. When the items is sold (s_{13}) the Goal to sell items is satisfied: $s_{13} \xrightarrow{\text{satisfy}} g_2$. Similarly, when the receipt is generated (s_{15}), the packages is prepared with the item and customer receipt (s_{16}), and the package is shipped to the customer address (s_{14}), the Goal to manage sold items is satisfied ($(s_{16} \wedge s_{15} \wedge s_{14}) \xrightarrow{\text{satisfy}} g_1$).

Nòmos 3 metamodel

The Nòmos 3 concepts are presented in the class diagram of figure 3.3.

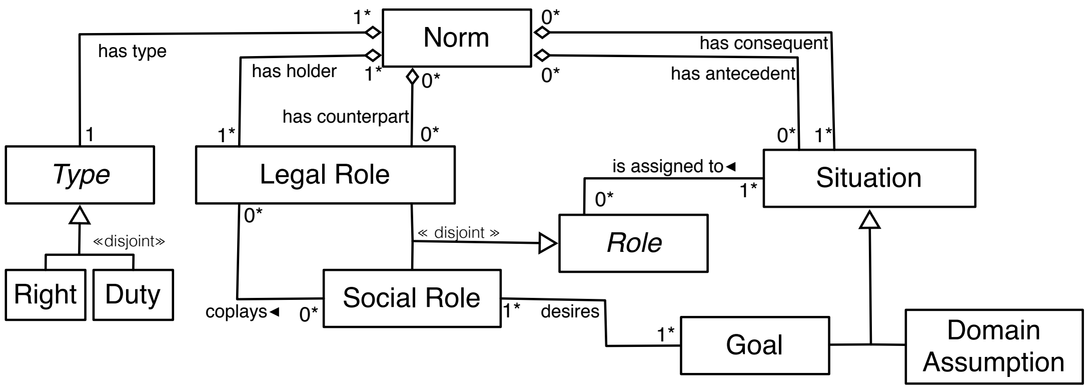


Figure 3.3: Class diagram of the Nòmos 3 concepts.

A **Norm** is composed of five elements: a Type, two Legal Roles, and two Situations.

- A Norm has exactly one Type, and a Type can type one or more Norms. With this latter choice, we are assuming that if a Type exists it is because there is at least one Norm with that type.

- A Norm has one or more holders (Legal Roles), and a Legal Role can be the holder of one or more Norms. With this latter choice, we assume that a Norm always has a holder, and also that if a Legal Role exists, then it is addressed by (and therefore the holder of) at least one Norm.

- A Norm can have zero or more counterparts (Legal Roles), and a Legal Role can be the counterpart of zero or more Norms. We assume that a Norm does not always has a counterpart, and also that if a Legal Role exists it is not necessarily the counterpart of a Norm.

- A Norm can have zero or more antecedents (Situations), and a Situation can be the antecedent of zero or more Norms. With this choices we assume that a Norm does not always has an antecedent, and also that if a Situation exists it is not necessarily the antecedent of a Norm. The former assumption is driven by the fact that there can be Norms that have no antecedent (e.g., "The right of a person to live"). The latter represents the fact that in our model Situations can be a Norm's antecedent, but they are not necessarily one.

- A Norm has one or more consequents (Situations), and a Situation can be the consequent of zero or more Norms. With this latter choice, we assume that a Norm always has a consequent, and also that if a Situation exists it is not necessarily the consequent of a Norm. The former assumption represents the fact that a Norm should always specify what needs to be done in order to comply with it. The latter represents the fact that in our model Situations can be a Norm's consequent, but they are not necessarily one.

Type is an abstract class classifying a Norm as either a Right or a Duty. Right or Duty are the two disjoint subtypes of Type class.

Role is another abstract class that represents the General concept of Role. In our domain we distinguish two types of Role: **Legal Role** and **Social Role**. These two classes are disjoint, so an instance of a Legal Role can not be also an instance of a Social Role.[4]

A Social Role can **coplay** zero or more Legal Roles, and a Legal Role can be coplayed by zero or more Social Roles. The former rule reflect the fact that in our domain there can be Social Roles who are not (significantly) characterized from a legal point of view. For example in the running example of the online seller and system to manage his invoices, the Social Role of

[4]Often the name that laws are often about social settings, such as marriage, handling of personal information etc.

`Online Seller` coplays a Legal Role significant in the context of the developed system, while the Social Role of `Sys Admin` for example would not coplay any Legal Role with respect to the norms about VAT invoices. The latter rule reflect the fact that, despite some Legal Roles may exist in the law, when they have no applicable norms it can be the case that they are not represented in the modeled domain. In the example of before, the law regulating VAT invoices may conceive the Legal Role of `VAT-invoice Controller` as the Legal Role who has to verify the invoices in case of a verification procedure. However, since the aim of the system is to support the `Online Seller`, the Social Role coplaying the Controller is not present in the modeled system.

A Social Role **desires** at least one or more Goals, and a Goal is desired by at least one Social Role. Both rules reflect two axioms of our domain: if a Social Role exists (and is modeled in the system) it must desire at least one goal, and he can desire multiple goals; if a Goal exists, it must be desired by at least one Social Role, and it can be desired by multiple Social Roles (e.g., in case of a delegation).

Goal and **Domain Assumption** are the two subclasses of **Situation**.

A Situation is assigned to zero or more Roles, and a Role is assigned one or more Situations. The first rule models the fact that a Situation does not have to be assigned to a Role (e.g. a Domain Assumption is a Situation but it is not assigned to a Role), and it can be assigned to multiple Roles (e.g., in case of a delegation). The second rule represent that a Role always has at least one assigned Situation: for Legal Roles, the Situation(s) satisfying one norm it is the holder of, for Social Role at least the Situation(s) assigned by the desired Goal.

3.3 Syntax and semantics

In the following we describe the syntax and semantics of the modeling language.

The syntax of the language is formed according to the BNF rules defined in this section. The semantics domain of Nòmos 2 includes satisfaction values, fulfillment value, and compliance value. Relationships in Nòmos 3 do not have a value, but are seeing as propagating value to their target in order to reason about the concepts in the model (see chapter 4).

Situations

Situations are states of the world represented by a proposition. We use Situation to represent the antecedent and consequent of a Norm, the responsibilities assigned to Roles, and the Situations that need to be satisfied for a Goal to be satisfied.

As a Situation is a proposition, and we use denote them by a subscripted s letter. Situations can be in conjunction or disjunction, and there can be situations which amount to a conjunction

x	y	$x \wedge y$	$x \vee y$
ST	ST	ST	ST
ST	SF	SF	ST
ST	SU	SU	ST
SF	ST	SF	ST
SF	SF	SF	SF
SF	SU	SF	SF
SU	ST	SU	ST
SU	SF	SF	SF
SU	SU	SU	SU

Table 3.1: Conjunction and disjunction for satisfaction values.

or a disjunction of others. Hence the following rules.

$$\text{Sit} ::= s_1 \mid s_2 \mid \ldots \tag{3.1}$$

$$\text{Sit}_\wedge ::= \wedge_{i=1..n, n>1} \text{Sit}_i \tag{3.2}$$

$$\text{Sit}_\vee ::= \vee_{i=1..n, n>1} \text{Sit}_i \tag{3.3}$$

$$S ::= \text{Sit} \mid \text{Sit}_\wedge \mid \text{Sit}_\vee \tag{3.4}$$

Satisfaction value. When there is evidence that a Situation is satisfied we say a Situation has value ST (Satisfaction True) and use the function $sat(x)$ to indicate the satisfiability of the Situation x: in this case, $sat(x) = \text{ST}$. When there is evidence that is not satisfied it has value SF (Satisfaction False), $sat(x) = \text{SF}$. When there is no evidence about the satisfaction of a Situation we say it has value SU (Satisfaction Undefined), $sat(x) = \text{SU}$. With the third value of undefined, we avoid assuming in our models that leaf nodes without values are, by default, satisfied or failed (similar to cite[Giorgini]). The satisfaction value that a Situation gets in a Nòmos 3 model depends on the values propagated by the relationships that it participates in. Multiple relationships targeting the same Situation are treated as being in disjunction and represent alternative ways to satisfy a Situation. The conjunction and disjunction satisfaction value of two Situations is summarized in table 3.1.

The value of the Situations in the model is either asserted before doing some kind of computation on the model, or is computed from values of other Situations in the model. By asserting a Situation's value, we mean that the modeller chooses a satisfaction value, and assigns it to a Situation (either directly or by including requirements satisfied by said Situations).

Norm

A Norm is a five tuple $norm = (type, holder, counterpart, antecedent, consequent)$.

type	antecedent	consequent	norm
Duty/Right	ST	ST	Compliant
Duty	ST	SF/SU	Violated
Right	ST	SF/SU	Tolerated
Duty/Right	SF	ST/SF/SU	Tolerated
Duty/Right	SU	ST/SF/SU	Inconclusive

Table 3.2: Rules for assigning values to Norms.

- the *type* of the Norm can be right or duty.
- the *holder* is the Legal Roles addressed by the Norm (e.g., the Physician) and who has to satisfy the Norm if it applies.
- the *count* is the optional Legal Role whose interests are helped if the Norm is satisfied (e.g., the Patient).
- the *antecedent/consequent* of a Norm are Situations to satisfy for the Norm to be applicable/satisfied

Norms are tuples of two Legal Roles (see eq. (3.9)), two Situations, and one Type:

$$N ::= (T, LegR, LegR, S, S) \tag{3.5}$$

$$T ::= right \mid duty \tag{3.6}$$

Compliance value. A Norm can be evaluated to compliant ($com(n)$), violated ($vio(n)$), tolerated ($tol(n)$), or inconclusive ($inc(n)$).

Differently from other concepts, the value of a Norm depends on the Norm's type, and its applicability and satisfiability (i.e., the values of the Norm's antecedent and consequent).

When a Norm's consequent is satisfied, we say that the *norm is satisfied*, and when a Norms's consequent is not satisfied (or has undefined satisfiability) we say that the Norm is not satisfied (has undefined satisfiability).

When a Norm's antecedent is satisfied, we say that the *norm is applicable*, and when a Norms's antecedent is not satisfied (or has undefined satisfiability) we say that the Norm is not applicable (has undefined applicability).

The rules to evaluate the compliance value of a Norm are summarized in table 3.2.

- A norm is *compliant* when it is applicable and satisfied.
- A norm is *violated* when it is a duty, is applicable and is not satisfied.

- A norm is *tolerated* when either it is an applicable right not satisfied (i.e., I have the right to do something but I'm not doing it), or when the norm (regardless from the type) is not applicable.
- A norm is *inconclusive* when it is unknown if it is applicable (regardless from the satisfaction of the consequent)

Multiple relationships targeting the same Norm are treated as being in disjunction and are considered alternative ways to make a Norm applicable/satisfied (according to the type of relation).[5]

Roles

In the Nòmos 3 language there are two Roles: a Legal Role is denoted by a subscripted lr, while a Social Role is denoted by a subscripted sr:

$$\begin{aligned} R &::= SocR \mid LegR & (3.7)\\ SocR &::= sr_1 \mid sr_2 \mid \ldots & (3.8)\\ LegR &::= lr_1 \mid lr_2 \mid \ldots & (3.9) \end{aligned}$$

Fulfillment value. In Nòmos 3 responsibilities of Roles are represented in terms of Situations assigned to the Role. The satisfaction values of Situations, which the Role is responsible for, determine the fulfilment value of the Role.

When there is evidence that a Role is fulfilled we say a Role has value FT (Fulfillment True) and use the shortcut function $ful(x)$ to indicate the fulfillment of a role $ful(x) = \text{FT}$, which is equivalent to indicating the positive satisfiability of the proposition associated to the role: $ful(x) = \text{FT} \equiv sat(x) = \text{ST}$. When there is evidence that is not fulfilled it has value FF (Fulfillment False), $ful(x) = \text{FF} \equiv sat(x) = \text{SF}$. When there is no evidence about the fulfillment of a Role we say it has value SU (Fulfillment Undefined), $ful(x) = \text{FU} \equiv sat(x) = \text{SU}$.

The fulfillment value of a Role depends on the semantics of the relationships that it is involved in. Legal Roles are responsible for the Situations satisfying the applicable norms they hold. The fulfillment value of a Legal Role is defined in the semantics of the relationship between a Legal Role and a Norm (relationship *holder* below). Social Roles have two responsibilities. The first is in the requirements model where they are responsible for the Situations satisfying the Goals they want (relationship *wanted*). The other responsibility is in the legal model where they are responsible for the Situations that are responsibility of the Legal Role they co-play (relationship

[5]The applicability and satisfiability values of a norm depend on the satisfaction value of the antecedent or consequent. Multiple relationships targeting a Norm are therefore treated as multiple relationships targeting a norms consequent (Situation) or antecedent (Situation). For example, if a Norm's antecedent receives from two different relationships value ST and SF, the two values are treated in disjunction according to table 3.1, and the final value of the antecedent is ST.

x	y	$x \wedge y$	$x \vee y$
FT	FT	FT	FT
FT	FF	FF	FT
FT	FU	FU	FT
FF	FT	FF	FT
FF	FF	FF	FF
FF	FU	FF	FF
FU	FT	FU	FT
FU	FF	FF	FF
FU	FU	FU	FU

Table 3.3: Conjunction and disjunction of fulfillment values.

coplays). The fulfillment value of a Social Role is defined in the semantics of the two relationships involving the Social Role (relationships *wanted* and *coplays* below).

Differently from Norms and Situations, multiple relationships targeting the same Role are treated as being in conjunction. The Role (Social or Legal) is fulfilled only if all the relationships targeting the Role propagate positive fulfillment. The conjunction and disjunction of fulfillment values follows the same logic introduced for satisfaction values, and is summarized in table 3.3.

Goals

Goals are Situations desired by a Social Role. The proposition used to represent the corresponding desired state of the world is denoted with a subscripted g:

$$Goal ::= g_1 \mid g_2 \mid \ldots \tag{3.10}$$

Similar to Situations, Goals can be in conjunction or disjunction with other Goals, and there can be Goals which amount to a conjunction or a disjunction of others

$$\begin{aligned} \text{Goal}_\wedge &::= \wedge_{i=1..n,n>1}\, \text{Goal}_i & (3.11)\\ \text{Goal}_\vee &::= \vee_{i=1..n,n>1}\, \text{Goal}_i & (3.12)\\ G &::= \text{Goal} \mid \text{Goal}_\wedge \mid \text{Goal}_\vee\,. & (3.13) \end{aligned}$$

Satisfaction value. Goals are Situations desired by at least one Social Roles ($goal(x) \rightarrow situation(x) \wedge \exists y : srole(y) \wedge want(y, x)$).

Goals have the same satisfaction values of Situations which are used to represent when the Goal is satisfied ST (Satisfaction True), not satisfied SF (Satisfaction False), or there is no evidence about its satisfaction SU (Satisfaction Undefined). Multiple relationships targeting the same Goal are treated as being in disjunction and represent alternative ways to satisfy a Goal.

$satisfy(x,y)$		$break(x,y)$		$satisfy^*(x,y)$		$break^*(x,y)$	
x	y	x	y	x	y	x	y
ST	ST	ST	SF	ST	SU	ST	SU
SF	SU	SF	SU	SF	ST	SF	SF
SU	SU	SU	SU	SU	SU	SU	SU

Table 3.4: Propagation rules for $satisfy/satisfy^*$ and $break/break^*$ relations.

The conjunction and disjunction satisfaction value of two Goal follows that defined for Situations (table 3.1).

Domain Assumption

A Domain Assumption (DA) is a Situation that is known to hold in the domain. The proposition used to represent the corresponding proposition is denoted with a subscripted *da*

$$DA \quad ::= \quad da_1 \mid da_2 \mid \ldots \tag{3.14}$$

Satisfaction value. DAs have the same satisfaction values of Situations. Since DA are known to hold in the domain, the value of a DAs is assumed to be satisfied ST (Satisfaction True), and is not computed from values of other Situations in the model.

Relationships

Relationships between Situations. The $satisfy$, $satisfy^*$, $break$ and $break^*$ are relationships between Situations, giving the rules Rel_{SS} below:

$$\text{Rel}_{SS} \quad ::= \quad S \xrightarrow{\text{satisfy}} \text{Sit} \mid S \xrightarrow{\text{break}} \text{Sit} \mid S \xrightarrow{\text{satisfy}*} \text{Sit} \mid S \xrightarrow{\text{break}*} \text{Sit}\,. \tag{3.15}$$

Table 3.4 defines how satisfiability values are propagated over the relation. The $break^*$ relationship is used to represent that if the source situation is not satisfied, then the target is not satisfied. Otherwise — if the source situation is not satisfied or has undefined satisfiability — nothing can be said about the satisfaction of the target.

The $satisfy$ relationship is used to represent that when the source situation is satisfied, the target is also satisfied. Otherwise — if the source situation is not satisfied or has undefined satisfiability — nothing can be said about the satisfaction of the target.

The $satisfy^*$ relationship is used to represent that if the source situation is not satisfied, then the target is satisfied. Otherwise — if the source situation is not satisfied or has undefined satisfiability — nothing can be said about the satisfaction of the target.

The $break$ relationship is used to represent that when the source situation is satisfied, the target is not satisfied. Otherwise — if the source situation is not satisfied or has undefined satisfiability — nothing can be said about the satisfaction of the target.

Conjunction and disjunction. Conjunction and disjunction of Situations can be used to express when two or more Situations need to be satisfied together or where at least one should be satisfied. Table 3.1 summarizes how satisfiability values are combined for conjunction and disjunction. For example, we have two situations s_1 and s_2 which satisfy in conjunction a third situation s_3 ($(s_1 \wedge s_2) \xrightarrow{\text{satisfy}} s_3$). Table 3.1 allows us to evaluate the value of the conjunction: for example if $sat(s_1) = ST$ and $sat(s_2) = SF$, the conjunction of the two values is SF. Table 3.4 allows us to evaluate the value propagated by the *satisfy* relation: since the source is SF, the target situation s_3 receives value SU.

Relationships between Situations and Norms. A Situation can activate, block, satisfy, or break a Norm. This gives the rule Rel_{SN} below:

$$\begin{aligned} \text{Rel}_{SN} \quad ::= \quad & S \xrightarrow{\text{activate}} N \mid S \xrightarrow{\text{block}} N \mid S \xrightarrow{\text{satisfy}} N \mid S \xrightarrow{\text{break}} N \\ & S \xrightarrow{\text{activate}*} N \mid S \xrightarrow{\text{block}*} N \mid S \xrightarrow{\text{satisfy}*} N \mid S \xrightarrow{\text{break}*} N \end{aligned} \tag{3.16}$$

The *activate*/*activate** relationship is used to propagate positive satisfaction to a Norm's antecedent when the source situation is satisfied/not-satisfied. It correspond to a *satisfy*/*satisfy** relationship from the source situation to the Situation representing the Norm's antecedent, and it follows the semantics of the corresponding relations.

The *block*/*block** relationship is used to propagate negative satisfaction to a Norm's antecedent when the source situation is satisfied/not-satisfied. It correspond to a *break*/*break** relationship from the source situation to the Situation representing the Norm's antecedent, and it follows the semantics of the corresponding relations.

The *satisfy*, *satisfy**, *break* and *break** are relationships between Situations and Norms, used to propagate satisfaction to a Norm's consequent according to the semantics of the relationship defined in table 3.4.

Relationships between Norms. The following rule is used to write them as formulas the three relationships between Norms:

$$\text{Rel}_{NN} \quad ::= \quad N \xrightarrow{\text{endorse}} N \mid N \xrightarrow{\text{derogate}} N \mid N \xrightarrow{\text{imply}} N \tag{3.17}$$

Relationships between Norms are defined using a combination of the four basic relationships above, and we therefore refer to them as shortcut relations.

- the **derogate** relationship is used to represent that when the source Norm is complied with, then the target Norm is not applicable. The semantics of the relationships can be derived by defining the relationship in terms of the four above. A derogate relationship $derogate(n_1, n_2)$ means that $\exists situation(s_1), situation(s_2)$ such that $activate(s_1, n_1)$, $satisfy(s_2, n_1)$ and $block(s_1 \wedge s_2, n_2)$.

holder(x, y)		*wanted*(x, y)		*coplay*(x, y)	
x	y	x	y	x	y
com/tol	FT	ST	FT	FT	FT
vio	FF	SF	FF	FF	FF
inc	FU	SU	FU	FU	FU

Table 3.5: Propagation rules for *holder*, *wanted* and *coplay* relations.

- the **endorse** relationship is used to represent that when the source Norm is complied with, then the target Norm is applicable. The semantics of the relationships can be derived by defining the relationship in terms of the four above. An endorse relationship $endorse(n_1, n_2)$ means that $\exists situation(s_1), situation(s_2)$ such that $activate(s_1, n_1)$, $satisfy(s_2, n_1)$ and $activate(s_1 \wedge s_2, n_2)$.

- the **imply** relationship is used to represent that when the source Norm is complied with, then the target Norm is also complied with. The semantics of the relationships can be derived by defining the relationship in terms of the four above. An imply relationship $imply(n_1, n_2)$ means that $\exists situation(s_1), situation(s_2)$ such that $activate(s_1, n_1)$, $satisfy(s_2, n_1)$ and $activate(s_1, n_2)$, $satisfy(s_2, n_2)$.

Relationship between Legal Roles and Norms. *holder* is a relationship between a Norm and a Legal Role, giving the following rule:

$$\text{Rel}_{RN} \quad ::= \quad N \xrightarrow{\text{holder}} LegR \tag{3.18}$$

This relationship represents that the Legal Role is responsible for the Norm, and is used to evaluate the fulfillment of a Legal Role. The Legal Role is responsible for the Situations satisfying the applicable Norms it holds. Since the compliance value of a Norm depends on the satisfiability of Situations (antecedent and consequent), the fulfillment of the Legal Role is determined by the Situations making the Norm applicable/satisfied. We use the compliance value of Norms to determines the fulfillment of the Legal Role.

Table 3.5 defines how fulfillment values are propagated over the relation. When the Norm is compliant or tolerated, the target Legal Role is fulfilled. If the Norm it holds is violated, the Legal Role is not fulfilled. Otherwise, if the Norm is inconclusive, then the Legal Role has undefined fulfillment The variability of how a given Legal Role is fulfilled, is given by the same variability underlying a Norm's applicability and satisfiability.

Multiple relationships targeting the same Role are treated in conjunction, so a Legal Role is considered fulfilled only when all the Norms it holds are compliant or tolerated.

Relationship between Social Roles and Goals. The relationship *wanted* is a relationship between a Goal and a Social Role:

$$\text{Rel}_{RG} \quad ::= \quad G \xrightarrow{\text{wants}} SocR \tag{3.19}$$

This relationship represents that the Social Role is responsible for the Situations satisfying the Goal it wants.

Table 3.5 defines how fulfillment values are propagated over the relation. The relationship is used to represent that when the Goal is satisfied the Social Role is fulfilled, when the Goal is not satisfied the Social Role is not fulfilled, otherwise if nothing can be said about the satisfaction of a Goal the Social Role has undefined fulfillment.

Multiple relationships targeting the same Role are treated in conjunction, so a Social Role who wants multiple Goals, it is considered fulfilled only when all the Goals he/she wants are achieved (see table 3.3 for the conjunction of fulfillment value).[6]
[7]

Relationship between Roles and Situations. The relationship *reserved* is a relationship between a Situation and a Social Role or a Legal Role:

$$\text{Rel}_{RS} \quad ::= \quad Sit \xrightarrow{\text{reserved}} SocR | Sit \xrightarrow{\text{reserved}} LegR \tag{3.20}$$

This relationship represents that a given Situation is *only* responsibility of a specific Role (Legal or Social). So when a Situation is *reserved* for a Role, it is important that the specified Role is the one entitled to bring about such Situation. This relationship does not propagate values and it will be used for reasoning in our models through schemas of relationships (see chapter 5).

Relationship between Roles. The relationship *coplay* is a relationship between a Social Role and a Legal Role:

$$\text{Rel}_{RR} \quad ::= \quad SocR \xrightarrow{\text{coplay}} LegR \tag{3.21}$$

This relationship represents that the Social Role is also responsible for the Situations that make the Legal Role fulfilled. So when a Social Role coplays a Legal Role, and the latter is not fulfilled, the former is also not fulfilled.

Table 3.5 shows how fulfillment values are propagated over the relation. Basically a Social Role has to be fulfilled from a requirements perspective (captured by the *want* relation) and from a legal perspective (captured by *coplay* relation). The relationships targeting the Social

[6]When the top Goal of a Social Role is operationalized, the *wanted* relationship is specified for the top Goal. The Social Role is fulfilled when the top goal is achieved, and in our models we assume that the lower level Goals are also *wanted* by the Social Role.

[7]When a top Goal is operationalized, we assume that the lower level Goals are also desired by the Social Role.

Role are treated as in conjunction, only when all relationships targeting a Social Role propagate positive fulfillment (all *wants* and all *coplays*), the Social Role is fulfilled.

The fulfillment of a Social Role can be summarized as follows. A Social Role is fulfilled when all the Goal he wants are satisfied and the Legal Roles he coplays are fulfilled. A Social Role is not fulfilled when either one Goal he wants is not satisfied or the Legal Role he coplays is not fulfilled: A Social Role has undefined fulfillment when either a Goal he wants has undefined satisfiability or a Legal Role he coplays is has undefined fulfillment.

The conjunction and disjunction of fulfillment values follows the same logic introduced for satisfaction values, and is summarized in .

Relationships among Situations, Goals and DA. The $satisfy/satisfy^*$ and $break/break^*$ are relationships among Situations, Goals and DAs, and are summarized in the rules as follows. A Situations can be satisfied/not-satisfied by a DA:

$$\mathrm{Rel}_{SGD} \quad ::= \quad DA \xrightarrow{\text{satisfy}} \mathrm{Sit} \mid DA \xrightarrow{\text{break}} \mathrm{Sit} \mid DA \xrightarrow{\text{satisfy}*} \mathrm{Sit} \mid DA \xrightarrow{\text{break}*} \mathrm{Sit} \tag{3.22}$$

A Goal can be satisfied/not-satisfied by a DA:

$$\mathrm{Rel}_{SGD} \quad ::= \quad \ldots \mid DA \xrightarrow{\text{satisfy}} \mathrm{Goal} \mid DA \xrightarrow{\text{break}} \mathrm{Goal} \mid DA \xrightarrow{\text{satisfy}*} \mathrm{Goal} \mid DA \xrightarrow{\text{break}*} \mathrm{Goal} \tag{3.23}$$

Goals can satisfy/not-satisfy a Situation, and Situations can satisfy/not-satisfy a Goal:

$$\mathrm{Rel}_{SGD} \quad ::= \quad \ldots \mid G \xrightarrow{\text{satisfy}} \mathrm{Sit} \mid G \xrightarrow{\text{break}} \mathrm{Sit} \mid G \xrightarrow{\text{satisfy}*} \mathrm{Sit} \mid G \xrightarrow{\text{break}*} \mathrm{Sit} \mid \tag{3.24}$$

$$G \xrightarrow{\text{satisfy}} G \mid G \xrightarrow{\text{break}} G \mid G \xrightarrow{\text{satisfy}*} G \mid G \xrightarrow{\text{break}*} G \mid \tag{3.25}$$

$$S \xrightarrow{\text{satisfy}} \mathrm{Goal} \mid S \xrightarrow{\text{break}} \mathrm{Goal} \mid S \xrightarrow{\text{satisfy}*} \mathrm{Goal} \mid S \xrightarrow{\text{break}*} \mathrm{Goal} \tag{3.26}$$

The propagation of satisfiability value over the relationship as well as their conjunction/disjunction, is the same as before (see tables 3.4 and 3.3).

Nomos model

Any Nòmos 3 model M is a set of formulas, $M = \{\phi_1, \ldots, \phi_n\}$ formed according to the BNF rules defined in this section and summarized in figure 3.4. Basically, a Nòmos 3 model is a set of Situations, Roles, Norms, and relationships over them, hence the following, and final rule.

$$\phi \quad ::= \quad S \mid R \mid N \mid \mathrm{Rel}\,. \tag{3.27}$$

$$
\begin{array}{rcl}
\mathrm{Sit} & ::= & s_1 \mid s_2 \mid \ldots \\
\mathrm{Sit}_\wedge & ::= & \wedge_{i=1..n,n>1}\, \mathrm{Sit}_i \\
\mathrm{Sit}_\vee & ::= & \vee_{i=1..n,n>1}\, \mathrm{Sit}_i \\
S & ::= & \mathrm{Sit} \mid \mathrm{Sit}_\wedge \mid \mathrm{Sit}_\vee \\
N & ::= & (T, \mathit{LegR}, \mathit{LegR}, S, S) \\
T & ::= & \mathit{right} \mid \mathit{duty} \\
R & ::= & \mathit{SocR} \mid \mathit{LegR} \\
\mathit{SocR} & ::= & sr_1 \mid sr_2 \mid \ldots \\
\mathit{LegR} & ::= & lr_1 \mid lr_2 \mid \ldots \\
\mathit{Goal} & ::= & g_1 \mid g_2 \mid \ldots \\
\mathrm{Goal}_\wedge & ::= & \wedge_{i=1..n,n>1}\, \mathrm{Goal}_i \\
\mathrm{Goal}_\vee & ::= & \vee_{i=1..n,n>1}\, \mathrm{Goal}_i \\
G & ::= & \mathrm{Goal} \mid \mathrm{Goal}_\wedge \mid \mathrm{Goal}_\vee \\
DA & ::= & da_1 \mid da_2 \mid \ldots \\
\mathrm{Rel}_{SS} & ::= & S \xrightarrow{\text{satisfy}} \mathrm{Sit} \mid S \xrightarrow{\text{break}} \mathrm{Sit} \mid S \xrightarrow{\text{satisfy}*} \mathrm{Sit} \mid S \xrightarrow{\text{break}*} \mathrm{Sit} \\
\mathrm{Rel}_{SN} & ::= & S \xrightarrow{\text{activate}} N \mid S \xrightarrow{\text{block}} N \mid S \xrightarrow{\text{satisfy}} N \mid S \xrightarrow{\text{break}} N \\
 & & S \xrightarrow{\text{activate}*} N \mid S \xrightarrow{\text{block}*} N \mid S \xrightarrow{\text{satisfy}*} N \mid S \xrightarrow{\text{break}*} N \\
\mathrm{Rel}_{NN} & ::= & N \xrightarrow{\text{endorse}} N \mid N \xrightarrow{\text{derogate}} N \mid N \xrightarrow{\text{imply}} N \\
\mathrm{Rel}_{RN} & ::= & N \xrightarrow{\text{holder}} \mathit{LegR} \\
\mathrm{Rel}_{RG} & ::= & G \xrightarrow{\text{wanted}} \mathit{SocR} \\
\mathrm{Rel}_{RS} & ::= & \mathit{Sit} \xrightarrow{\text{reserved}} \mathit{SocR} | \mathit{Sit} \xrightarrow{\text{reserved}} \mathit{LegR} \\
\mathrm{Rel}_{RR} & ::= & \mathit{SocR} \xrightarrow{\text{coplay}} \mathit{LegR} \\
\mathrm{Rel}_{SGD} & ::= & DA \xrightarrow{\text{satisfy}} \mathrm{Sit} \mid DA \xrightarrow{\text{break}} \mathrm{Sit} \mid DA \xrightarrow{\text{satisfy}*} \mathrm{Sit} \mid DA \xrightarrow{\text{break}*} \mathrm{Sit} \mid \\
 & & DA \xrightarrow{\text{satisfy}} \mathrm{Goal} \mid DA \xrightarrow{\text{break}} \mathrm{Goal} \mid DA \xrightarrow{\text{satisfy}*} \mathrm{Goal} \mid DA \xrightarrow{\text{break}*} \mathrm{Goal} \mid \\
 & & G \xrightarrow{\text{satisfy}} \mathrm{Sit} \mid G \xrightarrow{\text{break}} \mathrm{Sit} \mid G \xrightarrow{\text{satisfy}*} \mathrm{Sit} \mid G \xrightarrow{\text{break}*} \mathrm{Sit} \mid \\
 & & G \xrightarrow{\text{satisfy}} G \mid G \xrightarrow{\text{break}} G \mid G \xrightarrow{\text{satisfy}*} G \mid G \xrightarrow{\text{break}*} G \mid \\
 & & S \xrightarrow{\text{satisfy}} \mathrm{Goal} \mid S \xrightarrow{\text{break}} \mathrm{Goal} \mid S \xrightarrow{\text{satisfy}*} \mathrm{Goal} \mid S \xrightarrow{\text{break}*} \mathrm{Goal} \\
\mathrm{Rel} & ::= & \mathrm{Rel}_{SS} \mid \mathrm{Rel}_{SN} \mid \mathrm{Rel}_{NN} \mid \mathrm{Rel}_{RN} \mid \mathrm{Rel}_{RG} \mid \mathrm{Rel}_{RS} \mid \mathrm{Rel}_{RR} \mid \mathrm{Rel}_{SGD} \\
\phi & ::= & S \mid R \mid N \mid \mathrm{Rel}
\end{array}
$$

Figure 3.4: BNF rules defining the grammar of the Nòmos 3 language.

3.4 Chapter summary

In this chapter we have presented the first contribution of the thesis: the Nòmos 3 language for modeling laws and software requirements. First, we have introduced the concepts and relationships of the language (section 3.2) that allow the modeler to represent both the law and requirements. Then we have then presented the syntax and semantics of Nòmos 3 (section 3.3) and the rules to be used to create a Nòmos 3 model.

The main objective and novelty about the modeling language is that it allows the representation of the conditional mechanisms — applicability and satisfiability — that make a norm complied or not. Modeling this conditional mechanisms that underlines norms is of particular importance when modeling a method to ensure compliance of requirements. First it allows to reason about the different alternative ways that a norm can be complied with. Secondly, it is possible in the requirements model to reason over the effect of adding a new requirement or modifying it, and evaluate the legal consequences of such action. In the Nòmos 3 language, requirements are represented in terms of goals and the roles who wants them. Linking the roles in the requirements model to the roles in the law, allows to link the two models and evaluate which norms are applicable to a given role in the requirements model, and how to comply with it. The same conditionality mechanism allows to reason over goals and their compliance, as well as the compliance of the roles who desire such goals.

Chapter 4

Reasoning with variability

The aim of this chapter is to introduce the different type of variability reasoning that are possible with the Nòmos 3 modeling language. The semantics of the language allows the relationships in the model to act as a label-propagation mechanism. This mechanism is used to perform different type of reasoning and help the analyst answer different type of questions when evaluating the compliance of a set of requirements. In the legal model, the language allows to explore the variability behind a Norm's applicability and satisfiability: we investigate this aspect in the first part of the chapter titled 'Legal Variability' (section 4.1). In the second part of the chapter we investigate the variability behind a Role's fulfillment in both the legal and requirements model (section 4.2).

Reasoning in Nòmos 3 is performed by means of forward or backward reasoning algorithms. A Nòmos 3 model is basically an annotated AND/OR graph showing how elements are connected by the relationships in the model, an how satisfaction value is propagated throughout the model.

- *Forward reasoning* involves giving satisfiability values to some nodes in a Nòmos 3 network and then using the semantics of the relationships in the language to propagate labels to the other nodes of the network. Given an initial values assignment to situations (input situations) forward reasoning focuses on the forward propagation of these initial values to all other situations and to the norms/roles of the model according to activate the rules described in the previous chapter. Initial values represent the evidence available about the satisfaction of the situations, namely evidence about the state of the Situation. After the forward propagation of the initial values, the user can look the final values of the other nodes of interest. In other words, the user observes the effects of the initial situations over the model.

- *Backward reasoning* involves giving a desired value to some target nodes of a network and then searching back through the network for possible input values leading to the desired final value. For example, we set the desired final values of a target norm/goal or role, and we want to find possible initial assignments to the input situations, which would cause the desired final values of the target norm/goal/role by forward activate satisfy satisfy propagation. In other words, the user searches for the initial assignment to situation that would propagate to some nodes a desired value.

In a Nòmos 3 model we are interested in exploiting both types of algorithms in order to perform different type of reasoning to help the analyst deal with the alignment of a set of requirements with a piece of law. Elements (or nodes) in the Nòmos 3 model (network) correspond to one of the four Nòmos 3 concepts: Situation, Norms, Roles, Goals, and Domain Assumption. Relationships in the model propagate values according to the semantics of the language described in the previous chapter.
We associate to each situation/goal a variable $sat(s)$ with values ST, SF, SU, representing the current evidence of satisfiability of situation/goal s/g. Domain Assumption are always associated to a positive satisfiability value corresponding to the evidence that the assumption holds.
Each norm has a compliance value — $compl\_val(n_i)$ can be com, tol, vio, inc — and in the algorithms we use the primitive functions $antecedent(n_i)$, $consequent(n_i)$, $holder(n_i)$ to indicate the antecedent, consequent and holder of the Norm n_i. We refer to the satisfaction of the consequent of the norm as the satisfiability value of a Norm (values ST SF SU). The satisfaction of the antecedent of the norm, is the applicability value of a Norm: AT indicates that the antecedent of the norm is satisfied, AF that the antecedent is not satisfied, and AU that the antecedent has undefined applicability. The satisfiability value, the applicability value, and the norm type are used to evaluate the compliance value of the norm.
Roles are associated to the variable $ful(r)$ with values FT, FF, FU, representing the current evidence of fulfillment of the role r. For all elements the default values are SU, AU, and FU.

In this chapter we will use the primitive function $ForwardReasoning()$ to indicate the call to the function that performs the propagation of initial values $InitVal[node\_id, node\_value]$ in the model M according to the semantics of the language.[1] The output of the $ForwardReasoning()$ algorithm is the list of nodes with their final assignments to the nodes in the model $Value\_nodes\_Forward[node\_id, node\_value]$. We will use a simplified version of the algorithm for backward reasoning that generates assignments to situations, analyzes the model through forward reasoning, and returns the first solution returns the desired values in some target nodes $TargetVal[node\_id, node\_value]$.[2] For the purpose of this work, we assume that backward and forward reasoning algorithms also take care of possible cycles in graphs.

In the chapter we will use the scenario of the Online-seller/VAT-registered-business introduced in the visual notation of the language (section 3.2.3). The legal model depicts an excerpt of the British legislation on VAT record keeping and VAT invoices,[3] that requires VAT-registered

[1] $node\_id$ will follow the notation introduced in the syntax of the language, where norm are denoted with a subscribed n letter, situation with a subscribed s letter, roles with a subscribed r letter.

[2] For simplicity we assume that the function returns the first solution found, or an empty solution set when no solution is found.

[3] https://www.gov.uk/vat-record-keeping/vat-invoices.

business to issue a valid invoice for the good sold. The requirements model shows an excerpt of a goal model for a software to help online sellers keep track of goods and transactions.

4.1 Legal variability

One important purpose and characteristic of the Nòmos 3 language is to be able to model and analyze variability in laws: understanding when a norm is applicable or satisfied has important consequences on the design of methods for assuring the compliance of requirements to laws. Different requirements make applicable or satisfied different norms, and with the Nòmos 3 language we want to be able to reason about these two conditions in order to evaluate the Situations characterizing different aspects of the compliance of a Norm.

In this direction, forward and backward reasoning algorithms can be applied to Nòmos 3 legal models to answer several questions regarding norms and compliance:

1. Which Situations should be brought about in a Nòmos 3 legal model to make a set of Norms applicable/not-applicable?
2. Given an initial value to a set of Situations in a Nòmos 3 legal model, which Norms are applicable?
3. Which Situations should be brought about in a Nòmos 3 legal model to make a set of Norms satisfied/not-satisfied?
4. Given an initial value to a set of Situations in a Nòmos 3 legal model, which Norms are satisfied?
5. Which Situations should be brought about in a Nòmos 3 legal model to make a set of Norms complied/violated?
6. Given an initial value to a set of Situations in a Nòmos 3 legal model, which compliance value have the Norms?

In the following we introduce the different algorithms that allow the analyst to answer the questions above. The running scenario that will be used to illustrate the reasoning algorithms is the one of the VAT record keeping introduced in the previous chapter (see section 3.2.3).

4.1.1 Applicability search

Applicability Search is intended to find the set of situations that make a desired (sub)set of norms applicable/not-applicable. This algorithm is used to answer question 1 above: Which Situations should be brought about in a Nòmos 3 legal model to make a set of Norms applicable/not-applicable?

The input to our algorithm is the Nòmos 3 model M and a list of norm-nodes with the associated desired applicability value ($DesiredVal[norm\_id, app\_val]$). The output of the algorithm is a list describing the satisfiability value for the situations in the model $solution[sit\_id, sat\_val]$. The algorithm consists in a straightforward application of a backward reasoning algorithm that searches for assignments of values to Situations which would propagate the desired values to the target nodes. First it generates a tentative assignment to the situations in the model and propagates the value with a forward reasoning (line 3–4). Then it controls that for every norm in the model (line 5, $\forall norm(n_i)$ in M) the applicability value of the norm corresponds to the desired one (line 5, $tmp\_sol.n_i.app\_value() == DesiredVal.n_i.app\_value()$). If the condition holds for every norm, a solution has been found (line 7) and returned (line 11, 12), otherwise the loop is repeated. If no solution is found (line 13), the algorithm returns an empty array.

```
Input: $M$, $DesiredVal[norm\_id, app\_val]$
Output: $solution[sit\_id, sat\_val]$
1  boolean sol = false; solution[] = empty;
2  repeat
3      generate $assignment_i[sit\_id, sat\_val]$;
4      $tmp\_sol[] =$ ForwardReason($M, assignment_i$);
5      if $\forall norm(n_i)$ in $M$: $tmp\_sol.n_i.app\_value() == DesiredVal.n_i.app\_value()$;
6      then
7          sol=true
8      end
9  until sol=true or space alternatives explored;
10 if sol=true then
11     $solution[] = tmp\_sol.r_i[]$;
12     return solution[];
13 else
14     return solution[]
15 end
```

Algorithm 1: Applicability search (`applicability-search()`). This algorithm evaluates the satisfaction value that a set of Situations in a Nòmos 3 model (M) must have, in order to achieve the desired applicability value for a set of Norms ($DesiredVal[norm\_id, app\_val)$.

Example. In the running scenario, we consider the duty of the VAT-registered business to issue valid invoices for the good that is sold (figure 4.1). In this example, we use the applicability

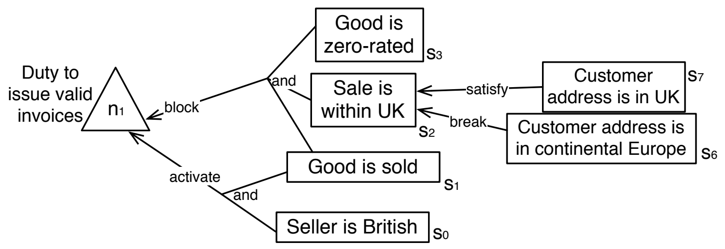


Figure 4.1: Applicability search/analysis: scenario of the online seller.

search to evaluate which Situations should be satisfied in order for the duty n_1 to be applicable. The algorithm generates a candidate solution and applies forward reasoning to evaluate if the applicability value that the norm obtains is the same of the desired one. For example, a candidate solution where s_0 has unknown of false satisfiability, while s_1, s_7, s_3 hold, would make the norm not applicable. When s_7 holds, then the relationship $satisfy$ propagates positive satisfiability to s_2 which also holds. Then, when all three Situations s_1, s_2, s_3 are satisfied their conjunction is also ST. Lastly, when the source is ST the relationship *block* propagates negative applicability AF to the target (i.e., negative satisfiability to the norm's antecedent), which makes the Norm not applicable. However a candidate solution where instead s_0, s_1 hold, while s_3 is false, would lead to the desired duty n_1 to be applicable: the conjunction of the satisfiability values of s_0, s_1 is ST, and the relationship *activate* propagates positive applicability AT to the target (i.e., positive satisfiability to the norm's antecedent), which makes the norm applicable.

4.1.2 Applicability analysis

Applicability Analysis is intended to find the applicability value of a set of norms, given a set of situations with an initial assignment. This algorithm is used to answer question 2 above: Given an initial value to a set of Situations in a Nòmos 3 legal model, which Norms are applicable/not-applicable?

The input to our algorithm is the Nòmos 3 model M and a list of situations-nodes with their initial satisfaction value ($InitialAss[sit\_id, sat\_val]$). The output of the algorithm is a list describing the applicability of each norm $solution[norm\_id, app\_val]$. The algorithm consists in just a call to the forward reasoning algorithm that implements the propagation rules that allow nodes to propagate values across the model, and returns the list of norms of the model (line 2) with their applicability value returned by the reasoning.

```
   Input: $M$, $InitialAss[sit\_id, sat\_val$
   Output: $solution[norm\_id, app\_val]$
1  $output\_forward[node, final\_val]$ = ForwardReason($M$,$InitialAss$);
2  foreach $i$ in $output\_forward[]$ such that ($node_i$.type == norm) do
3      solution.add($node_i$, $node_i.app\_val()$ );
4  end
5  return solution[];
```

Algorithm 2: Applicability analysis (`applicability-analysis()`). This algorithm returns the applicability value of the norm in a Nòmos 3 model (M), given an initial satisfiability value to the Situations in the model ($initialAss[sit\_id, sat\_val]$).

Example. In the running scenario illustrated in figure 4.1 the algorithm for the applicability analysis is given as input the Nòmos 3 model in figure and a list of situation with the corresponding satisfaction value. We will consider the following satisfaction value for situations: $sat(s_0, s_1, s_3)$ = ST, $sat(s_7, s_6)$ = SU. The first two situations s_0, s_1 are both satisfied: their conjunction also computes ST and the *activate* relationship propagates positive applicability AT to the target (i.e., positive satisfiability to the norm's antecedent). Since both s_6 and s_7 are SU, both relationships which target s_2 propagate SU. Since multiple relationships targeting the same situation are treated as begin in disjunction, the disjunction of the two propagated value is SU. This propagation allows to evaluate to SU the satisfiability of s_2. Despite s_1, s_3 are ST, the conjunction of s_1, s_2, s_3 results in SU. The *block* relationship therefore propagates unknown applicability AU to the target (i.e., unknown satisfiability to the norm's antecedent). The two relationships targeting the norm are treated as being in disjunction: *activate* propagates AT, *block* propagates AU, and the disjunction of the two values is AT (i.e., the disjunction of the two satisfaction values). The duty n_1 is therefore evaluated as applicable.

4.1.3 Satisfiability search

Satisfiability Search is intended to find the set of situations that make a desired (sub)set of norms satisfied/not-satisfied. This algorithm is used to answer question 3 above: Which Situations should be brought about in a Nòmos 3 legal model to make a set of Norms satisfied/not-satisfied?

The input to our algorithm is the Nòmos 3 model M and a list of norm-nodes with the associated desired satisfiability value ($DesiredVal[norm\_id, sat\_val]$). The output of the algorithm is a list describing the satisfiability value for the situations in the model $solution[sit\_id, sat\_val]$. The algorithm consists in a straightforward application of a backward reasoning algorithm that searches for assignments of values to Situations which would propagate the desired values to the target nodes. First it generates a tentative assignment to the situations in the model and propagates the value with a forward reasoning (line 3–4). Then it controls that for every norm in the model (line 5, $\forall norm(n_i)$ in M) the satisfiability value of the norm corresponds to the desired one (line 5, $tmp\_sol.n_i.sat\_value() == DesiredVal.n_i.sat\_value()$). If the condition holds for every norm, a solution has been found (line 7) and returned (line 11, 12), otherwise the loop is repeated. If no solution is found (line 13), the algorithm returns an empty array.

Example. The running scenario illustrated in figure 4.2 shows the small Nòmos 3 model in input to the algorithm, and the desired value for the target norm n_i is ST. The algorithm generates candidates solutions and applies forward reasoning to evaluate if the satisfiability value that the norm obtains is the same of the desired one. For example, a candidate solution where

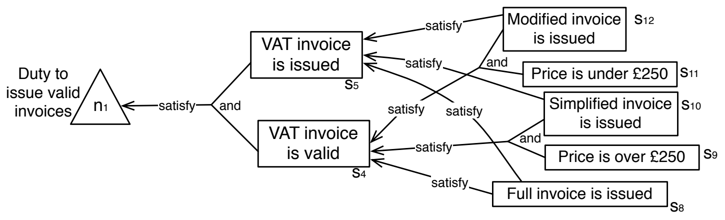


Figure 4.2: Satisfiability search/analysis: scenario of the online seller.

```
   Input: $M$, $DesiredVal[norm\_id, sat\_val]$
   Output: $solution[sit\_id, sat\_val]$
1  boolean sol = false; solution[] = empty;
2  repeat
3      generate $assignment_i[sit\_id, sat\_val]$;
4      $tmp\_sol[]$ = ForwardReason($M, assignment_i$);
5      if $\forall norm(n_i)$ in $M$: $tmp\_sol.n_i.sat\_value() == DesiredVal.n_i.sat\_value()$;
6      then
7          sol=true
8      end
9  until sol=true or space alternatives explored;
10 if sol=true then
11     $solution[] = tmp\_sol.r_i[]$;
12     return solution[];
13 else
14     return $solution[]$
15 end
```

Algorithm 3: Satisfiability search (`satisfiability-search()`). This algorithm evaluates the satisfaction value that a set of Situations in a Nòmos 3 model (M) must have, in order to achieve the desired satisfiability value of a set of Norms ($DesiredVal[norm\_id, sat\_val]$).

$sat(s_8)$ = SF, $sat(s_9, s_{11})$ = ST, and the other situations have unknown satisfiability, would make the norm not satisfied. The conjunction between s_9 and s_{10} results in SU — same as between s_{11} and s_{12} — so all three relationships targeting s_4 propagate SU. A disjunction of three SU values results in s_4 being evaluated to SU. Similarly, all the relationships targeting s_5 propagate SU, which in disjunction result in s_5 begin evaluated to SU. Both s_4 and s_5 are SU, their conjunction is SU, so in turns the $satisfy$ relationship targeting the norm propagates unknown satisfiability SU to the duty. For example, a candidate solution where $sat(s_8)$ = ST, and all the other situations have unknown satisfiability, would make the norm satisfied. s_8 propagates ST to both s_4 and s_5. Both situations receives unknown satisfiability from the other incoming relations. The disjunction between ST, SU, SU is ST, so both s_4 and s_5 obtain value ST. Their conjunction is evaluated to ST, which in turns propagates positive satisfiability ST to the duty.

4.1.4 Satisfiability analysis

Satisfiability Analysis is intended to find the satisfaction value of a set of norms, given a set of situations with an initial assignment. This algorithm is used to answer question 4 above: Given an initial value to a set of Situations in a Nòmos 3 legal model, which Norms are satisfied/not-satisfied?

The input to our algorithm is the Nòmos 3 model M and a list of situations-nodes with their initial satisfaction value ($InitialAss[sit\_id, sat\_val]$). The output of the algorithm is a list describing the satisfiability of each norm $solution[norm\_id, sat\_val]$. The algorithm consists in a call to the forward reasoning algorithm that implements the propagation rules that allow nodes to propagate values across the model, and returns the list of norms of the model (line 2) with their satisfiability value returned by the reasoning.

```
Input: $M$, $InitialAss[sit\_id, sat\_val$
Output: $solution[norm\_id, sat\_val]$
1 $output\_forward[node, final\_val]$ = ForwardReason($M$,$InitialAss$);
2 foreach $i$ in $output\_forward[]$ such that ($node_i$.type == norm) do
3     solution.add($node_i$, $node_i.sat\_val()$ );
4 end
5 return solution[];
```

Algorithm 4: Satisfiability analysis (`satisfiability-analysis()`). This algorithm evaluates the satisfiability value of the norms in a Nòmos 3 model (M), given an initial satisfiability value to the Situations in the model ($InitialAss[sit\_id, sat\_val]$).

Example. In the running scenario illustrated in figure 4.2 the algorithm for the satisfiability analysis is given as input the Nòmos 3 model in figure and a list of situation with the corresponding satisfaction value. We will consider the following satisfaction value for situations: $sat(s_8)$ = SF, $sat(s_9, s_{10})$ = ST, and $sat(s_{11}, s_{12})$ = SU. Since s_8 is SF, both the $satisfy$ relationships originating from this situation propagate value SU to their target. The conjunction of s_9, s_{10} is ST since both Situations hold, therefore s_4 receives ST as propagated value from the relationship $satisfy$ generated from the two relations. Through the direct $satisfy$ relation, s_{10} propagates ST to s_5. Lastly since s_{11} and s_{12} have unknown satisfiability, all the relationships originating from them and their conjunction propagate SU. Situation s_4 receives as values SU, ST, SU, and their conjunction results in s_4 being evaluated to ST. Situation s_5 receives the same values from the incoming relationships and is therefore also evaluated to ST. The $satisfy$ relationship from s_4 and s_5 has as source value the conjunction of the two ST values — evaluated to ST — and therefore propagates positive satisfaction ST to the norm.

4.1.5 Simple compliance analysis

Simple compliance analysis is intended to find the compliance value of a set of Norms, given a set of situations with an initial assignment. This algorithm is used to answer question 6 above: Given an initial value to a set of Situations in a Nòmos 3 legal model, which compliance value have the Norms?

The input to our algorithm is the Nòmos 3 model M and a list of situations with their initial satisfaction value ($InitialAss[sit\_id, sat\_val$). The output of the algorithm is a list describing the compliance value of each norm $solution[norm\_id, compl\_val]$. The algorithm consists in

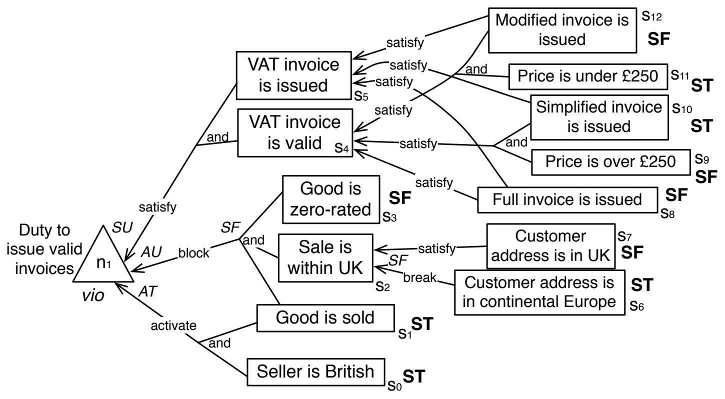


Figure 4.3: Simple compliance search/analysis: scenario of the online seller.

a straightforward application of the forward reasoning algorithm, and evaluation of the norm compliance value based on its satisfiability and applicability value. If a norm is applicable (line 4) and is satisfied (line 5), then it has value *com* (line 6). If it is applicable but not satisfied and is a duty (line 8) then it has value *vio* (line 9), otherwise it is a right and has value *tol* (line 11). When a norm has unknown applicability (line 12), then regardless from its type and its satisfaction value the norm is evaluated to *inc*. Lastly — a norm that is neither applicable AT nor unknown applicability SU (line 14) — when a norm is not applicable it has value *tol* regardless from its type and satisfaction value.

```
   Input: $M$, $InitialAss[sit\_id, sat\_val]$
   Output: $solution[norm\_id, compl\_val]$
1  assignment-sat[$norm\_id$, $sat\_value$] = satisfiability-analysis($M$,$InitialAss$);
2  assignment-app[$norm\_id$, $app\_value$] = applicability-analysis($M$,$InitialAss$);
3  foreach $n_i \in M$ do
4      if assignment-app.value($n_i$) = AT then
5          if assignment-sat.value($n_i$) = ST then
6              solution.add( $n_i$, com );
7          else
8              if $n_i$.type = duty then
9                  solution.add( $n_i$, vio );
10             else
11                 solution.add( $n_i$, tol );
12     if assignment-app.value($n_i$) = AU then
13         solution.add($n_i$, inc);
14     else
15         solution.add( $n_i$, tol );
16 end
17 return solution;
```

Algorithm 5: Simple compliance analysis (`simple-compl-analysis()`). This algorithm evaluates the compliance value of a set of Norms in a Nòmos 3 model (M), with respect to an initial assignment to a set of situations ($InitialAss[sit\_id, sat\_val]$).

Example. The running scenario illustrated in figure 4.3 shows the small Nòmos 3 model in input to the algorithm, and computes the compliance value for the norm given an initial assignment of the Situations. As scenario we will consider that a British seller sells to a customer in France an item for £120 and issues a simplified invoice. The scenario corresponds to $s_0, s_1, s_6, s_{10}, s_{11}$ being ST and the others being SF. Since s_0 and s_1 are holding, the *activate* relationship propagates positive applicability AT to the norm. The customer address is in continental Europe, the relationship *break* between s_6 and s_2 propagates negative satisfiability SF to the target situation s_2. The conjunction between s_1 = ST, s_2 = SF and s_3 = SF is evaluated to SF, and the *block* relationship propagates unknown applicability AU to the norm. The final applicability value is then computed by the disjunction of the two values received (AT $\vee$ AU, i.e. satisfiability ST $\vee$ SU for the antecedent), which makes the duty applicable. The relationships originating in s_8 propagate SU to the target situations s_4 and s_5. The conjunction between s_9 = SF and s_{10} = ST is SF, so the $satisfy$ relationship targeting s_4 propagates SU. The conjunction between s_{11} = ST and s_{12} = SF is SF, so the other $satisfy$ relationship targeting s_4 also propagates SU. The $satisfy$ relationship from s_{10} targeting s_5 propagates ST, while the relationship from s_{12} = SF targeting s_5 propagates SU. The situation s_4 receives three SU values, which in disjunction make s_4 = SU. The situation s_5 receives SU ST and SU values, which in disjunction make s_5 = ST. The conjunction between s_4 and s_5 is SU, which in turns makes the duty not satisfied. The norm n_1 of type duty, that is applicable and not satisfied is evaluated to *vio*.

4.1.6 Simple compliance search

Simple compliance search is intended to find the satisfaction value that a set of Situations in a Nòmos 3 legal model must have in order for a set of Norms to be evaluated with a desired compliance value. This algorithm is used to answer question 5 above: Which Situations should be brought about in a Nòmos 3 legal model to make a set of Norms complied/violated/tolerated/inconclusive?

The input to our algorithm is the Nòmos 3 model M and a list of Norms with the associated desired compliance value ($DesiredVal[norm\_id, comp\_val]$). The output of the algorithm is a list describing the satisfiability value for the situations in the model $solution[sit\_id, sat\_val]$. The algorithm consists in a straightforward application of a backward reasoning algorithm that searches for assignments of values to Situations which would propagate the desired values to the target nodes. First it generates a tentative assignment to the situations in the model (line 3) and uses the simple-compl-analysis() algorithm to evaluate the compliance value of the norm with that given assignment (line 4). Then it controls that the satisfiability values of all norms correspond to the desired ones (line 5): if so, a solution has been found and returned (line 7, 11, 12), otherwise the loop is repeated. If no solutions are found (line 13), the algorithm returns an empty array.

```
Input: $M$, $DesiredVal[norm\_id, comp\_val]$
Output: $solution[sit\_id, sat\_val]$
boolean sol = false; solution[] = empty;
repeat
    generate $assignment_i[sit\_id, sat\_val]$;
    $tmp\_sol[norm\_id, compl\_val]$ = simple-compl-analysis($M, assignment_i$);
    if $\forall n_i$ in $M$: $tmp\_sol.n_i.comp\_value() == DesiredVal.n_i.comp\_value()$;
    then
        sol=true
    end
until $sol{=}true$ or space alternatives explored;
if $sol{=}true$ then
    $solution[] = tmp\_sol.r_i[]$;
    return solution[];
else
    return $solution[]$
end
```

Algorithm 6: Simple compliance search (`simple-compl-search()`). This algorithm evaluates the satisfaction value that a set of Situations in a Nòmos 3 model (M) must have, in order to achieve the desired compliance value of a set of Norms ($DesiredVal[norm\_id, comp\_val]$).

Example. The running scenario illustrated in figure 4.3 shows the small Nòmos 3 model in input to the algorithm, and the desired value for the target norm n_i is *com*.

The algorithm generates candidates solutions and applies forward reasoning to evaluate the applicability and satisfiability value that the norm obtains, and check that it corresponds to the desired one. For example, a candidate solution where $sat(s_1, s_7, s_3)$ = ST, and the other situations have unknown satisfiability, would make the norm tolerated. The three known situations make the norm not applicable, and the unknown situations make the norm not satisfied. A norm of type duty, that is not applicable and has unknown satisfiability has value tolerated *tol*. For example, a candidate solution where $sat(s_0, s_1, s_8)$ = ST, while the other situations have unknown satisfiability, would make the norm complied. The first two situations make the norm applicable, and the last one satisfies both s_4 and s_5 which in turn make the norm satisfied. A norm of type duty, that is applicable and satisfied has value complied *com*.

4.2 Roles fulfillment variability

An important factor in evaluating the compliance of requirements to a norm comes from the analysis of who must comply with it, which is just as important as what compliance entail. The Nòmos 3 modeling language is able to model and analyze the responsibilities of roles in both the legal and requirements model. By designing system roles and deciding which requirements they are responsible for, the analyst implicitly defines to which legal roles are involved. With the Nòmos 3 language we want to be able to reason about these responsibilities and evaluate if the roles are fulfilled.

Forward and backward reasoning algorithms can be applied to Nòmos 3 models to answer several questions regarding Roles:

1. Which Situations should be brought about in a Nòmos 3 model to make a set of Roles (Legal or Social) fulfilled/not-fulfilled?
2. Given an initial value to a set of Situations in a Nòmos 3 model, which Roles are fulfilled?
3. Given an initial value to a set of Situations in a Nòmos 3 model, which are the Legal Roles with applicable norms?
4. Given an initial value to a set of Situations in a Nòmos 3 model, to which Norms must some Legal Roles comply with?
5. Given an initial value to a set of Situations in a Nòmos 3 model, which are the Social Roles who have applicable norms?
6. Given an initial value to a set of Situations in a Nòmos 3 model, to which Norms must some Social Roles comply with?

In the following we introduce the different algorithms that allow the modeller to answer the questions above. The running scenario that will be used to illustrate the reasoning algorithms is the same as in the previous section.

4.2.1 Roles fulfillment search

Roles fulfillment search (Legal/Social Role fulfillment search) is intended to find the set of situations that make a desired set of legal/social roles fulfilled/not-fulfilled. This algorithm is used to answer the question 1 above: Which Situations should be brought about in a Nòmos 3 model to make a set of Roles fulfilled/not-fulfilled?

The input to our algorithm is the Nòmos 3 model M and a list of roles with the associated desired fulfillment value ($desiredFulfillment[role, fulfillment\_value]$). The output of the algorithm is a list describing the satisfiability value for the Situations in the model

$solution[sit\_id, sat\_val]$. The algorithm consists in a straightforward application of a backward reasoning algorithm that searches for assignments of values to Situations which would propagate the desired values to the target nodes. First it generates a tentative assignment to the situations in the model and propagates the value with a forward reasoning (line 3–4). Then it controls that the fulfillment values of all roles correspond to the desired ones (line 5): if so, a solution has been found and returned (line 7, 11, 12), otherwise the loop is repeated. If no solutions are found (line 13), the algorithm returns an empty array.

```
   Input: $M$, $desiredFulfillment[role, fulfillment\_value]$
   Output: $solution[sit\_id, sat\_val]$
 1 boolean sol = false; solution[] = empty;
 2 repeat
 3     generate $assignment_i[sit\_id, sat\_val]$;
 4     $tmp\_sol[]$ = ForwardReason($M, assignment_i$);
 5     if $\forall role(r_i)$ in $M$: $tmp\_sol.r_i.value() == desiredFulfillment.r_i.value()$;
 6     then
 7         sol=true
 8     end
 9 until sol=true or space alternatives explored;
10 if sol=true then
11     $solution[] = tmp\_sol.r_i[]$;
12     return solution[];
13 else
14     return $solution[]$
15 end
```

Algorithm 7: Role fulfillment search (role-fulfill-search()). This algorithm evaluates the satisfaction value that a set of Situations in a Nòmos 3 model (M) must have, in order to achieve the desired fulfillment value of a set of Roles (Legal/Social) ($desiredFulfillment[role, fulfillment\_value]$).

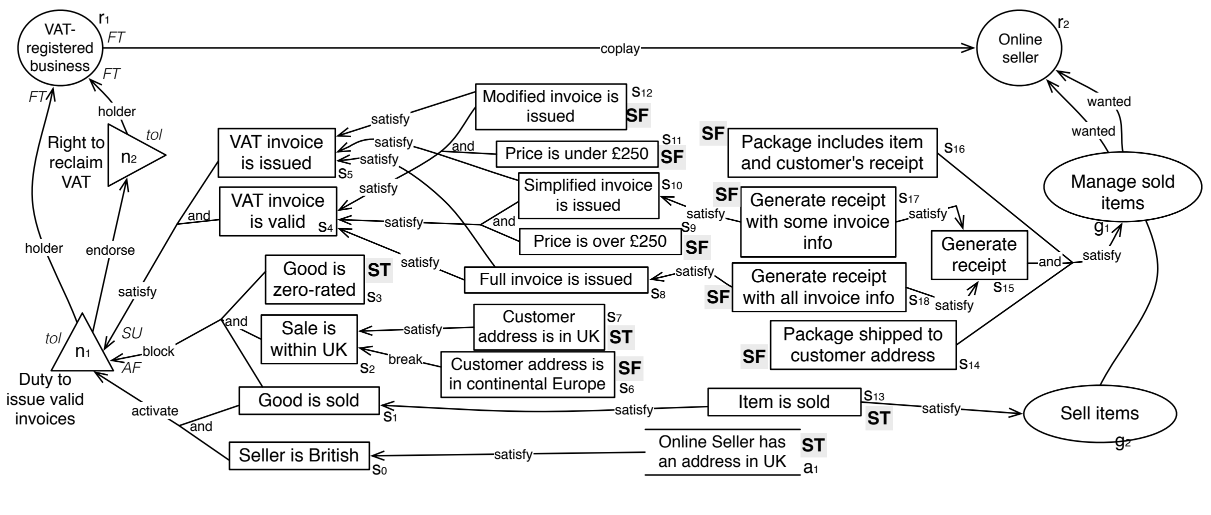


Figure 4.4: Role fulfillment search: scenario of the online seller.

Example. The running scenario illustrated in figure 4.4 considers the legal role r_1 of `VAT-registered business` and the social role r_2 of `Online-seller`. In this example we use the role-fulfill-search() algorithm to evaluate which Situations should be satisfied in order for the legal role r_1 to be fulfilled FT. The algorithm generates a candidate solution and applies forward reasoning to evaluate if the fulfillment value that the role obtains is the desired one. For example, a candidate solution where the domain assumption holds (a_1 = ST) and a zero-rated item is sold to a customer in U.K. ($sat(s_{13}, s_7, s_3)$ = ST, and the other situations are SF), would make the first norm n_1 tolerated (a duty not applicable) and the second norm n_2 would also be tolerated (right not applicable). The *holder* relationship propagates FT to the legal role in both cases. The final fulfillment value of the role is the conjunction of the two FT, and is therefore FT.

If we want to use the role-fulfill-search() algorithm to evaluate which situations should be satisfied in order for the social role r_2 to be fulfilled FT, the same procedure evaluates also which goals wanted by the social role are satisfied. When the same candidate solution as before is evaluated, while goal g_2 "Sell Items" is satisfied by the situation "Item is sold", goal g_1 is not satisfied as none of the situations targeting it are satisfied ($sat(s_{16}, s_{14})$ = SF and $sat(s_{15})$ = SU). So the two *wanted* relationships targeting r_2, propagate FU (from g_1) and FT (from g_2). Now the relationships targeting the social role are treated as being in conjunction, so the conjunction of the fulfillment value of the two *wanted* relationships and of the *coplay* relationship linking the social and legal role results in the social role having undefined fulfillment (FT ∧ FU ∧ FT = FU).

4.2.2 Role fulfillment analysis

Role fulfillment analysis is intended to find the fulfillment value of a set of Roles, given a set of situations with an initial assignment. This algorithm is used to answer question 2 above: Given an initial value to a set of Situations in a Nòmos 3 model, which Roles are fulfilled?

The input to our algorithm is the Nòmos 3 model M and a list of situation-nodes with their initial satisfaction value ($InitialAss[sit\_id, sat\_val]$). The output of the algorithm is a list describing the fulfillment of each norm $solution[role\_id, ful\_val]$. The algorithm consists in just a call to the forward reasoning algorithm that implements the propagation rules that allow nodes to propagate values across the model, and returns the list of roles of the model (line 2) with their fulfillment value returned by the reasoning.

```
Input: $M$, $InitialAss[sit\_id, sat\_val$
Output: $solution[role\_id, ful\_val]$
1 $output\_forward[node, final\_val]$ = ForwardReason($M$,$InitialAss$);
2 foreach $i$ in $output\_forward[]$ such that ($node_i$.type == role) do
3     solution.add($node_i$, $node_i.ful\_val()$ );
4 end
5 return solution[];
```

Algorithm 8: Role fulfillment analysis (`role-fulfill-analysis()`). This algorithm evaluates the fulfillment value of the roles in a Nòmos 3 model (M), given an initial satisfiability value to the Situations in the model ($InitialAss[sit\_id, sat\_val]$).

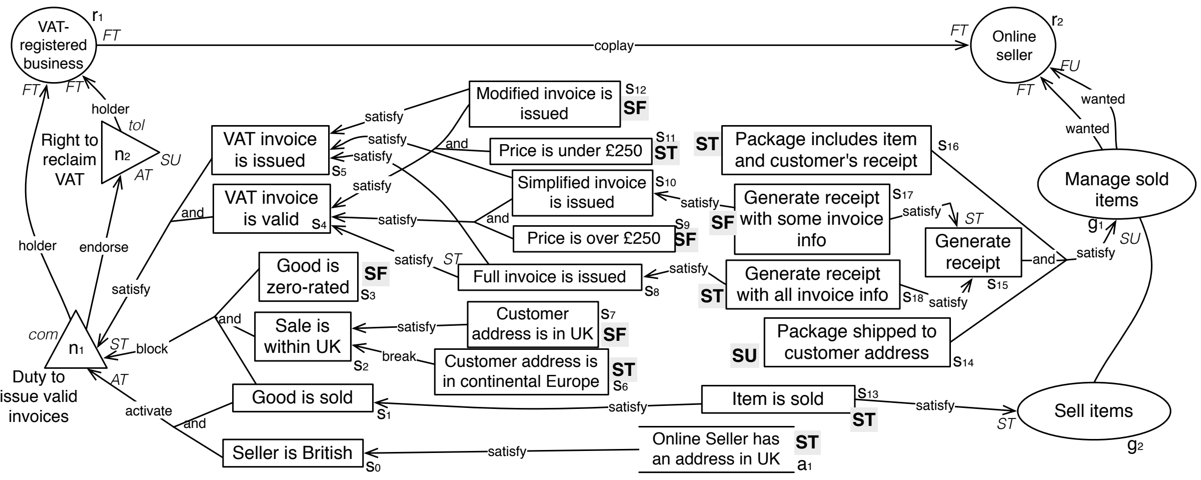


Figure 4.5: Role fulfillment analysis: scenario of the online seller.

Example. In the running scenario illustrated in figure 4.5 the algorithm for the fulfillment analysis is given as input the Nòmos 3 model in figure and a list of situation with the corresponding satisfaction value. We will consider the following satisfaction value for situations (marked in bold in figure 4.5): $sat(a_1, s_{13}, s_6, s_{18}, s_{16}, s_{11})$ = ST, $sat(s_3, s_7, s_9, s_{17}, s_{12})$ = SF, and $sat(s_{14})$ = SU. The first two situations s_0, s_1 are both satisfied: the first by the Domain Assumption a_1, the second by the situation s_{13}. The conjunction of s_0 and s_1 also computes ST and therefore makes the norm n_1 applicable (the *block* relationship propagates unknown applicability from the conjunction SU of the three situation). Situation s_{18} makes s_8 satisfied, which in turn makes s_4 and s_5 satisfied, propagating positive satisfaction ST to the norm n_1 which is evaluated to compliant *com*. Since n_2 is complied, the relationship *endorse* makes n_2 applicable and — assuming unknown satisfiability — n_2 is evaluated to tolerated *tol*. Through the *holder* relationship both norms propagate positive fulfillment FT to the role r_1, which in conjunction make FT the final fulfillment value for the role r_1. Moreover, s_{13} positively satisfies the goal g_2, which in turn propagates positive fulfillment FT to the social role r_2 through the *wanted* relation. The other goal g_1 is instead evaluated to unknown satisfiability: while the receipt is generated ($sat(s_{15} = \text{ST})$ since s_{18} is satisfied) and the package includes the receipt ($sat(s_{16} = \text{ST})$), in the scenario it is not known whether the package has been shipped ($sat(s_{14} = \text{SU})$). The conjunction between the three situations is SU, which makes the $satisfy$ relationship propagate SU to the goal g_1 which propagates FU to the role r_2 through the *wanted* relation. The social role r_2 receives two fulfillment values (FU, FT) from the two wanted goals, and also receives positive fulfillment FU from the *coplay* relationship with the legal role r_1. The conjunction of the fulfillment values ($\text{FT} \wedge \text{FU} \wedge \text{FT} = \text{FU}$) makes the social role have unknown fulfillment FU. The algorithm returns for both roles, the corresponding fulfillment value: r_1, FT and r_2, FU.

4.2.3 Analysis of Norms applicable to Legal Roles

This algorithm is used to find the Legal Roles in a model that have applicable Norms, given a set of situations with an initial assignment. This algorithm is used to answer questions 3 and 4 above: Given an initial value to a set of Situations in a Nòmos 3 model, which are the Legal Roles with applicable norms? Given an initial value to a set of Situations in a Nòmos 3 model, to which Norms must the Legal Roles comply with?

The input to our algorithm is the Nòmos 3 model M and a list of situation-nodes with their initial satisfaction value ($InitialAss[sit\_id, sat\_val]$). The output of the algorithm is a list of legal roles together with the list of norms applicable to each legal role $solution[lrole\_id, norm\_appl[]]$. First the algorithm evaluates the applicability value for the norms (line 1) using the algorithm `applicability-analysis()`. Then for every legal role in the model, it selects the norms that have that legal role as holder (line 2-3). If the norm is applicable, it is added to the list of applicable norms for that role (line 4-5).[4] The algorithm then returns the solution with the list of legal roles with some applicable norms, and the list of applicable norms for each role.

To answer question 3 ("which Legal Roles have applicable Norms?"), only the list of Roles is considered from the solution set, while to answer question 4 ("to which Norms some the Legal Roles comply with?"), the complete solution with roles id and applicable norms

[4]We are assuming that the add functions adds a value to the list of norms if one already exits.

($solution[lrole\_id, norm\_appl[]]$) is considered.

```
Input: $M$, $InitialAss[sit\_id, sat\_val]$
Output: $solution[lrole\_id, norm\_appl[]$
assignment-app[$norm\_id$, $app\_value$] = applicability-analysis($M$,$InitialAss$);
foreach legal role $r_i$ in $M$ do
    foreach norm $n_i$ in $M$ such that $n_i.holder() == r_i$ do
        if assignment-app($n_i$)==$AT$ then
            solution.add($r_i$,$n_i$);
        end
    end
end
return solution[];
```

Algorithm 9: Noms applicable to legal role analysis (`norms-applicable-lrole()`). This algorithm evaluates the legal roles in a Nòmos 3 model (M) that are the holder of some applicable norms, given an initial satisfiability value to the Situations in the model ($InitialAss[sit\_id, sat\_val]$).

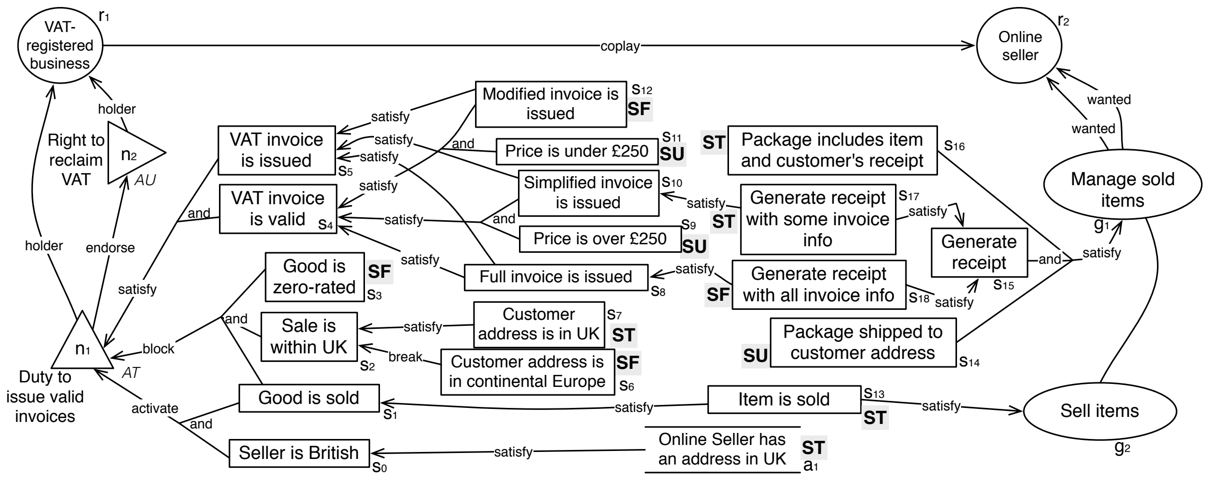


Figure 4.6: Noms applicable to legal role analysis: scenario of the online seller.

Example. In the running example illustrated in figure 4.7 we consider the Nòmos 3 model as input to the algorithm, together with the list of situations with satisfaction value corresponding to the following scenario: a British online seller has sold an item to a British customer ($sat(a_1, s_{13}, s_7)$ = ST, $sat(s_6, s_3)$ = SF), the item had an unknown price ($sat(s_9, s_{11})$ = SU); the seller has prepared a package containing the item and a receipt with some invoice information ($sat(s_{16}, s_{17})$ = ST, $sat(s_{18})$ = SF); it is not known whether the package has been shipped ($sat(s_{14})$ = SU). The algorithm first evaluates the applicability value for the norms: since a good is sold from a British seller (s_0 and s_1 are satisfied by a_1 and $s_1$3), the *activate* relationship propagate positive applicability AT to the norm n_1. The *block* relationship propagates unknown applicability AU (SF $\wedge$ ST $\wedge$ SF = SU, SU $\xrightarrow{block}$ AU) which in disjunction with AT, makes the duty n_1 to issue a valid invoice applicable. The *endorse* relationship does not propagate any value to n_2 which therefore remains with unknown applicability AU. Then the algorithm selects the legal role who is the holder of the norms (r_1) and adds the applicable norms to the solution list for that role. The output of the algorithm is the list of legal roles together with the list of norms applicable: in the considered scenario the legal role of `VAT-registered business` has one applicable norm (r_1, n_1).

4.2.4 Analysis of Norms applicable to Social Roles

This algorithm is used to find the Social Roles in a model that have applicable Norms, given a set of situations with an initial assignment. This algorithm is used to answer questions 5 and 6 above: Given an initial value to a set of Situations in a Nòmos 3 model, which are the Social Roles who have applicable norms? Given an initial value to a set of Situations in a Nòmos 3 model, to which Norms must some Social Roles comply with?

The input to our algorithm is the Nòmos 3 model M and a list of situation-nodes with their initial satisfaction value ($InitialAss[sit\_id, sat\_val]$). The output of the algorithm is a list of social role together with the list of norms applicable to each social role $solution[srole\_id, norm\_appl[]]$. First the algorithm evaluates the norm applicable to the legal roles (line 1) using the algorithm `applicability-analysis()`. Then for every social role in the model, it selects the legal roles that are coplayed by that social role (line 2-3). The social role has applicable all the norms applicable to each legal role coplayed (line 4). The algorithm returns the list of social roles with their applicable norms.

Similarly to the previous algorithm, to answer question 5 ("which Social Roles have applicable Norms?"), only the list of Roles is considered from the solution set, while to answer question 6 ("to which Norms some the Social Roles comply with?"), the complete solution with roles id and applicable norms ($solution[srole\_id, norm\_appl[]]$) is considered.

```
Input: $M$, $InitialAss[sit\_id, sat\_val]$
Output: $solution[role\_id, norm\_appl[]$
1 lrole-norm[$lrole\_id$, $norm\_appl[]$] = norms-applicable-lrole($M$, $InitialAss[sit\_id, sat\_val]$);
2 foreach social role $sr_i$ in $M$ do
3     foreach $r_i$ in $M$ such that $\exists coplay(sr_i, r_i)$ in $M$ do
4         solution.add($sr_i$,lrole-norm.$r_i$.norms[]);
5     end
6 end
7 return solution[];
```

Algorithm 10: Noms applicable to social role analysis (`norms-applicable-srole()`). This algorithm evaluates the social roles in a Nòmos 3 model (M) that coplay a legal role holder of some applicable norms, given an initial satisfiability value to the Situations in the model ($InitialAss[sit\_id, sat\_val]$).

Example. In the running example illustrated in figure 4.7 we consider the Nòmos 3 model as input to the algorithm, together with the list of situations with satisfaction value corresponding

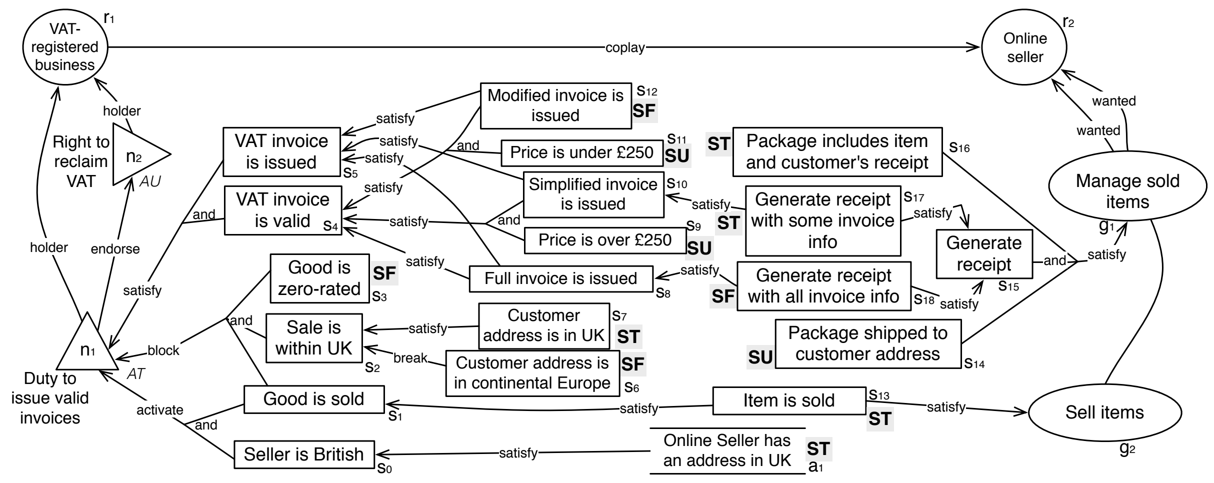


Figure 4.7: Noms applicable to legal role analysis: scenario of the online seller.

to the same scenario as in the previous algorithm: a British online seller has sold an item to a British customer ($sat(a_1, s_{13}, s_7)$ = ST, $sat(s_6, s_3)$ = SF), the item had an unknown price ($sat(s_9, s_{11})$ = SU); the seller has prepared a package containing the item and a receipt with some invoice information ($sat(s_{16}, s_{17})$ = ST, $sat(s_{18})$ = SF); it is not known whether the package has been shipped ($sat(s_{14})$ = SU). The algorithm first evaluates the norms applicable to the legal roles (r_1, n_1). Then it selects the social roles who coplay that legal role (only r_2 in this example), and it returns the list of social roles together with the list of norms applicable to the legal roles coplayed: (r_2, n_1).

4.3 Conclusions

Nòmos 3 models can be analyzed by means of backward and forward reasoning algorithm. Given an initial values assignment to situations, forward reasoning focuses on the forward propagation of these initial values to all other situations and to the norms of the model according to the semantics of the relationships they participate in. With backward reasoning, we set the desired final values of some target norms, and we search for possible initial assignments to the input situations.

The relationships in the model to act as a label-propagation mechanism used to perform different type of reasoning and help the analyst answer different type of questions when evaluating the compliance of a set of requirements. Tables 4.1 and 4.2 summarizes the algorithms that allow the modeler reason about Nòmos 3 model and answer specific questions about them. In table 4.1 we summarized the questions and algorithm helping the analyst investigate the variability behind a Norm's applicability and satisfiability (Legal Variability). In table 4.2 we summarized the questions and algorithm helping the analyst investigate a Role's fulfillment in both the legal and

Question	Reasoning about Legal Variability	
Which Situations should be brought about in a Nòmos 3 legal model to make a set of Norms applicable/not-applicable?	**Applicability Search**	**Input** M, $DesiredVal[norm\_id, app\_val]$
		Output $solution[sit\_id, sat\_val]$
Given an initial value to a set of Situations in a Nòmos 3 legal model, which Norms are applicable?	**Applicability Analysis**	**Input** M, $InitialAss[sit\_id, sat\_val$
		Output $solution[norm\_id, app\_val]$
Which Situations should be brought about in a Nòmos 3 legal model to make a set of Norms satisfied/not-satisfied?	**Satisfiability Search**	**Input** M, $DesiredVal[norm\_id, sat\_val]$
		Output $solution[sit\_id, sat\_val]$
Given an initial value to a set of Situations in a Nòmos 3 legal model, which Norms are satisfied?	**Satisfiability Analysis**	**Input** M, $InitialAss[sit\_id, sat\_val$
		Output $solution[norm\_id, sat\_val]$
Which Situations should be brought about in a Nòmos 3 legal model to make a set of Norms complied/violated?	**Simple Compliance Search**	**Input** M, $DesiredVal[norm\_id, comp\_val]$
		Output $solution[sit\_id, sat\_val]$
Given an initial value to a set of Situations in a Nòmos 3 legal model, which compliance value have the Norms?	**Simple Compliance Analysis**	**Input** M, $InitialAss[sit\_id, sat\_val]$
		Output $solution[norm\_id, compl\_val]$

Table 4.1: Algorithms for reasoning about the variability of law in a Nòmos 3 model.

requirements model (Role Variability).

Question	**Reasoning about Role Variability**	
Which Situations should be brought about in a Nòmos 3 model to make a set of Roles (Legal or Social) fulfilled/not-fulfilled?	**Roles fulfillment search**	**Input** M, $desiredFulfillment[role, fulf\_value]$
		Output $solution[sit\_id, sat\_val]$
Given an initial value to a set of Situations in a Nòmos 3 model, which Roles are fulfilled?	**Roles fulfillment analysis**	**Input** M, $InitialAss[sit\_id, sat\_val$
		Output $solution[role\_id, ful\_val]$
Given an initial value to a set of Situations in a Nòmos 3 model, which are the Legal Roles with applicable norms? *Given an initial value to a set of Situations in a Nòmos 3 model, to which Norms must some Legal Roles comply with?*	**Noms applicable to legal role analysis**	**Input** M, $InitialAss[sit\_id, sat\_val]$
		Output $solution[lrole\_id, norm\_appl[]$
Given an initial value to a set of Situations in a Nòmos 3 model, which are the Social Roles who have applicable norms? *Given an initial value to a set of Situations in a Nòmos 3 model, to which Norms must some Social Roles comply with?*	**Noms applicable to social role analysis**	**Input** M, $InitialAss[sit\_id, sat\_val]$
		Output $solution[role\_id, norm\_appl[]$

Table 4.2: Algorithms for reasoning about the variability of Role fulfillment in a Nòmos 3 model.

Chapter 5

Reasoning with violations and goals

The aim of this chapter is to introduce a new mechanism to detect the violation of a norm, and reasoning over the overall compliance of a set of goals. In chapter 4 we have seen how the Nòmos 3 modeling language allows to reason about the compliance of norms and fulfillment of roles, based on the situations that are brought about. However, reasoning over a norms' violation and compliance depends on three factors: (i) the situations that need to hold to make a norm applicable, (ii) the situations that need to hold to actually comply with the norm, and lastly (iii) the roles that bring about such situations.

In this chapter we first analyze the problem of violations that come about from roles who are not entitled to bring about some situations. Secondly, we expand the reasoning over a norm violation/compliance to include also the new mechanism and we illustrate the algorithms to evaluate the compliance of a set of goals with a given law.

5.1 Role-based violations

Today's enterprises critically depend on software systems for their daily operations. However, this requires that designers also assign the right responsibilities to the right roles, and evaluate who is entitled to perform some actions. For example, an electronic voting systems must support not only the function of voting through an electronic device, but also to ensure conformance to regulations concerning voting rights and obligations.

For instance, in a system for managing patients' health in an hospital, assigning the responsibility of prescribing drugs to the right role (e.g., the Attending Physician) is crucial: the same action performed by the wrong person (e.g., the Nurse) may not only endanger the patients' health but also make the hospital liable for the consequences. Furthermore even though the Attending Physician has final responsibility, legally and otherwise, for patient care[1] — he may want to delegate to another role in the system some of his tasks: prescribing drugs, changing

[1] https://en.wikipedia.org/wiki/Attending_physician.

therapy, update the patient medical record, and other medical activities. Evaluating whether a Secretary, Nurse or Physician Assistant[2] is entitled to perform one of those actions is again of crucial importance for the patients care and avoid violating laws about Medical Practice or Privacy.

In the context of this illustrative example we want help the analyst with the following questions: Is this role (e.g., the Nurse) entitled to achieve a given goal (e.g., prescribing drugs or printing a chart)? Moreover, if the system-to-be we are designing allows a Physician to delegate to the Nurse the prescription of a drug, does this delegation makes the designed system violate some norms?

In this section we address these two questions. First we introduce a new mechanism to evaluate violation of a norm in case the wrong role brings about these situations. With this extension, we allow the analyst to reason about compliance not only in terms of the situations brought about, but also for the role who is responsible for them. Secondly we introduce the concept of legally-valid social delegation and use it to evaluate whether the delegation of a responsibility in the system makes it in violations with some norms.

5.1.1 The reserve schema

The concepts and relationships of Nòmos 3 have been defined with the purpose of capturing the most relevant semantics constructs that can be found in the law, and in a GORE requirements model. During the requirements engineering process, a key activity consists in assigning the right responsibilities to the right actors: for example if the system-to-be must support the creation modification and deletion of a patients medical records, it is legally of critical importance to decide *who*, in a given hospital, is entitled to perform that actions. Indeed the right action done by the wrong person may lead to a violation of the patient's rights.

It is therefore extremely important to identify these cases of "right-action-wrong-role", and support the corresponding analysis in the models. To achieve such objective we verify a schema of relationships in the Nòmos 3 model. A schema identifies a specific set of Nòmos 3 concepts and relationships between them — similar to a path in a graph representing Nòmos 3 relationships as edges and concepts as nodes.

The core idea behind the "right-action-wrong-role" is that if in the law it is specified that a given responsibility is assigned only to a specific Legal Role, then in the requirements model we must check that the corresponding Social Role is also the only responsible.

In terms of modelling constructs, the clause that reserves a responsibility (Situation) to a Legal Role is represented in Nòmos 3 by the *reserved* relation. So if in a Nòmos 3 model there is the *reserved* relationship from a Situation to a Legal Role — e.g. if a Norm says that only the Attending Physician can prescribe drugs — it is necessary that in the requirements model:

[2]A healthcare professional who is licensed to practice medicine, https://en.wikipedia.org/wiki/Physician_assistant.

(a) there is a *reserved* relationship from that Situation to the Social Role who co-plays that Legal Role — e.g., the Social Role of Doctor-in-charge (who coplays the Legal Role of Attending Physician) must also be the only one who can prescribe drugs;

(b) and that such responsibility is operationalized for the correct Social Role — e.g., the Social Role of Doctor-in-charge is the Role with the goal of prescribing drugs.

The relationships and concepts used to express this restriction define the **reserve schema**.

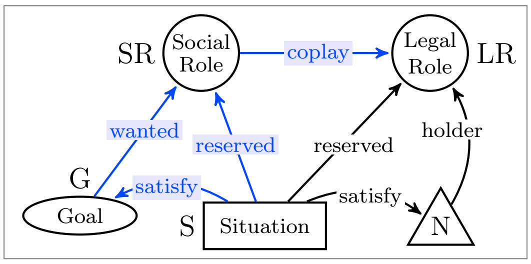


Figure 5.1: The concepts and relationships defining the reserve schema.

Figure 5.1 depicts the schema and highlights the key concepts and relations. In the model of the law it is specified that only a given Legal Role can bring about a Situation satisfying the Norm: this is what makes the schema applicable. The four highlighted relationships represent the constructs that have to be present in the requirements model to ensure compliance to the fragment of law.

The reserve schema in a Nòmos 3 model is defined as follows:

The reserve schema: *if* a *reserved* relationship is present in the legal model (S $\xrightarrow{\text{reserved}}$ LR), it is necessary that in the requirements model:

(a) the Social Role who *coplays* that Legal Role (SR $\xrightarrow{\text{coplay}}$ LR) has also *reserved* the same Situation (S $\xrightarrow{\text{reserved}}$ SR), and

(b) this Social Role wants a Goal (G $\xrightarrow{\text{wanted}}$ SR) that is satisfied by said Situation (S $\xrightarrow{\text{satisfy}}$ G).

This corresponds to ensuring that if a restriction is mandated by the law, (a) it is assigned to the right Social Role and (b) is correctly operationalized in the requirements model.

The following equation formally represents the formula reserve schema and its conditions. For the purpose of correctly representing the schema in a formal way we express the schema using FOL-style predicates.

$$\underbrace{(reserved(x, lr_1))}_{\substack{\text{condition triggering the}\\\text{schema to be applicable}}} \rightarrow \underbrace{\big(coplay(sr_2, lr_1) \wedge reserved(x, sr_2)\big)}_{\text{(a) first condition to be met}} \wedge \big(\underbrace{satisfy(x, g) \wedge wanted(g, sr_2)}_{\text{(b) second condition to be met}}\big) \tag{5.1}$$

Equation (5.1) can be read as follows as an *if-then* block. The left side of the implication represents the *if* condition and if that holds, *than* the right side should hold too (i.e., the left side represents the condition that makes the schema to be applicable): if a Situation x is reserved ($reserved(x, lr_1)$) for a Legal Role (lr_1), then the right side of the implication must hold. The right side represents the two conditions and that must hold:

(a) first a Social Role (sr_2) who coplays that Legal Role ($coplay(sr_2, lr_1)$) has the same Situation reserved ($reserved(x, sr_2)$);

(b) secondly the Goal g wanted by that Social Role is satisfied by the Situation ($satisfy(x, g)$).

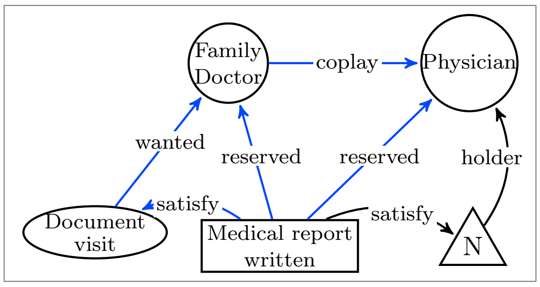


Figure 5.2: Reserve schema: example.

Figure 5.2 illustrates the schema with an extract from the simplified example of a software for managing hospital records and medical visits. A Norm requires that only the `Physician` is allowed to write a medical report about a visit. So when designing the system it is important to ensure that (a) only the Social Role `Family Doctor` — who coplays the Legal Role `Physician` — is the only one who is allowed to document a medical visit. This restriction is then (b) correctly operationalized as the `Family Doctor` wants a Goal ("Document visit") that is satisfied by the Situation reserved for him/her.

What triggers the schema to be applicable is the *reserved* relationship in the legal model: when there is that relationship in the model, it is necessary that the two other conditions hold. If the schema applies and there is the *reserved* relation, but the two conditions in the requirements model are not met — in other words, the "unique" responsibilities of a Legal Role were not represented and operationalized in the requirements of the system-to-be — the result is that the design is lacking some important information regarding the restriction, which leads to non-compliance of the system.

5.1.2 Violations of Norms with the reserve schema

Violations of the schema identify cases where the schema must apply — i.e. there is restriction in the legal model — but the conditions of the reserve schema do not hold — i.e. in the requirements model not all relationships are present.

In table 5.1 we have represented the five cases summarizing the truth table of the reserve schema formula, representing when the schema leads to a violation or not. In the first four cases the condition applies (i.e., it is True that there is the *reserved* relationship in the legal model) and the four logical combination are considered for the two conditions on the right side. In the last case, the condition does not apply (i.e., it is False that there is the *reserved* relationship in the legal model), so regardless from the value on the right side of the implication, the result will always hold.

In the following we analyze these 5 cases. We will illustrate the schema violations continuing with the previous simplified example of a software for managing hospital records and medical visits. Moreover, we will provide possible mitigations strategies for the violations and describe

Reserve Schema:			
$\underbrace{reserved(x, lr_1)}_{\text{condition}} \rightarrow$	$\underbrace{(coplay(sr_2, lr_1) \wedge reserved(x, sr_2))}_{\text{(a)}} \wedge$	$\underbrace{(satisfy(x, g) \wedge wanted(g, sr_2))}_{\text{(b)}}$	**Case:**
T	T	T	**1:** no violation
T	F	T	**2:** violation
T	T	F	**3:** violation
T	F	F	**4:** violation
F	*	*	**5:** no violation

Table 5.1: The truth table for the formula representing the reserve schema.

them using simple STRIPS rules Fikes and Nilsson [1971].[3]

Case 1: No violation

Reserve Schema:			
$\underbrace{reserved(x, lr_1)}_{\text{condition}} \rightarrow$	$\underbrace{coplay(sr_2, lr_1) \wedge reserved(x, sr_2)}_{\text{(a)}} \wedge$	$\underbrace{satisfy(x, g) \wedge wanted(g, sr_2)}_{\text{(b)}}$	**Case:**
T	T	T	**1:** no violation

Table 5.2: Case 1: the reserve schema applies and both conditions are met.

Case 1 represents the standard case used to introduce the reserve schema where all conditions are met. Figure 5.2 showed how in the legal model the presence of the *reserved* relationship — only a `Physician` can write a medical report — made the schema applicable. So when both conditions (a) and (b) hold, the formula or the reserve schema is true, and we are in a case where no violation is detected.

Case 2: Violation

Reserve Schema:			
$\underbrace{reserved(x, lr_1)}_{\text{condition}} \rightarrow$	$\underbrace{(coplay(sr_2, lr_1) \wedge reserved(x, sr_2))}_{\text{(a)}} \wedge$	$\underbrace{(satisfy(x, g) \wedge wanted(g, sr_2))}_{\text{(b)}}$	**Case:**
T	F	T	**2:** violation
	$(\neg\mathbf{coplay}(\mathbf{sr_2}, \mathbf{lr_1}) \wedge reserved(x, sr_2))$		**2a**
	$(coplay(sr_2, lr_1) \wedge \neg\mathbf{reserved}(\mathbf{x}, \mathbf{sr_2}))$		**2b**
	$(\neg\mathbf{coplay}(\mathbf{sr_2}, \mathbf{lr_1}) \wedge \neg\mathbf{reserved}(\mathbf{x}, \mathbf{sr_2}))$		**2c**

Table 5.3: Case 2: the reserve schema applies and condition (a) is not met.

[3]See also, `https://en.wikipedia.org/wiki/STRIPS`.

Case 2 represents the case where the reserve schema is applicable (there is a *reserved* relationship in the legal model) but one of the two condition — condition (a) — is not met. Condition (a) is false in three cases which are summarized in table 5.3: either the *coplay* relationship is missing, the *reserved* relationship is missing, or both are.

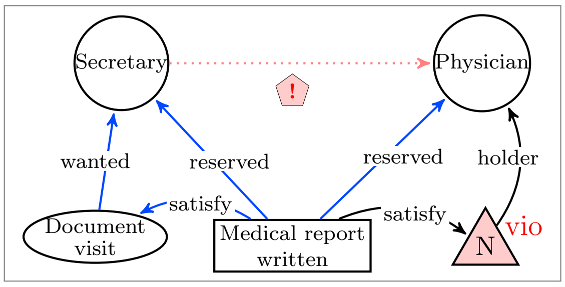


Figure 5.3: Schema violation 2a.

Schema violation 2a. This type of violation of the schema occurs when the *coplay* relationship linking the Social and Legal Role is missing ($\neg coplay(sr_2, lr_1)$ in table 5.3). The scenario is that of a Situation correctly *reserved* for a Legal and a Social Role, but the Situation is reserved and operationalized for the wrong Social Role. In the example of figure 5.3, in the requirements model the Situation "Medical report written" is *reserved* but for a Social Role that does not coplay the `Physician` (the `Secretary`), so the Physician's duty is evaluated to 'violation' as long as this responsibility is operationalized for the wrong role (and it is the wrong role because it does not coplay the Legal Role).

Mitigation Techniques. For this type of violation we distinguish two cases.

1. The first technique is based on the assumption that the Social Role sr_2 can coplay that Legal Role (consider figure 5.3 where the Social Role is `Family Doctor`). The simple rule is to add the missing *coplay* relation.

   Mitigation 1: `add:` $coplay(sr_2, lr_1)$

2. The second technique is based on the assumption that the Social Role sr_2 *can not* coplay that Legal Role (as represented in figure 5.3). This rule involves removing the *wanted* relationship from the first Social Role who does not coplay the Legal Role (first line in the mitigation rule), and then adding the Goal and reserved relationship to the Social Role who actually coplays the Legal Role lr_1 of `Physician` ($sr_3 s.t.\ coplay(sr_3, lr_1)$).

   Mitigation 2: `delete:` $wanted(g, sr_2) \wedge reserved(x, sr_2)$;
   `add:` $wanted(g, sr_3) \wedge reserved(x, sr_3)$ for $sr_3 s.t.\ coplay(sr_3, lr_1)$;

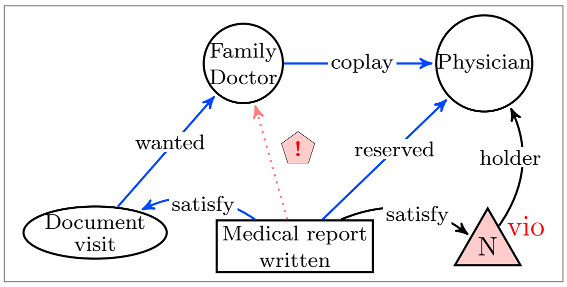


Figure 5.4: Schema violation 2b.

Schema violation 2b. This type of violation of the schema arises when the *reserved* relationship is missing in the requirements model ($\neg reserved(x, sr_2)$ in table 5.3). This case corresponds to the scenario where a Norm mandates that only a specific LegalRole can bring about a Situations, and this restriction is not represented in the requirements model. The Norm satisfied by said Situation is then evaluated as 'violation'. In the example of figure 5.5 when in the requirements model it is not represented that only the `Family Doctor` is allowed to write a medical report (dotted red line in figure 5.5), the Physician's duty is violated because the schema is not respected.

Mitigation Techniques. For this type of violation, correctly including in the requirements model the missing relationship is sufficient to resolve the problem.

Mitigation: `add:` $reserved(x, sr_2)$

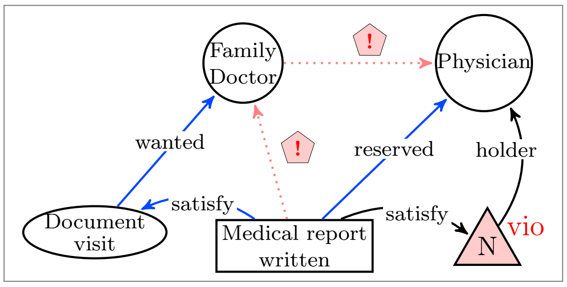


Figure 5.5: Schema violation 2c.

Schema violation 2c. This violation of the schema arises when both relationships in the (a) condition are missing ($\neg coplay(sr_2, lr_1) \wedge \neg reserved(x, sr_2)$ in table 5.3). This case corresponds to the combination of the two scenarios before. A Norm mandates that only a specific LegalRole can bring about a Situations, and in the requirements model: neither the restriction is represented, nor it is known whether the goal is operationalized for the right role.

In the example of figure 5.5, first in the requirements model it is not represented that only the `Family Doctor` is allowed to write a medical report (dotted red line in figure 5.5). Secondly it is not known/represented whether the Social Role of `Family Doctor` coplays the `Physician`.[4]

Mitigation Techniques. Since this type of violation arises as the combination of the two violations before, any conjunction of the mitigation techniques for violation 2a and 2b addresses the problem.

Case 3: Violation

Reserve Schema: $\underbrace{reserved(x, lr_1)}_{\text{condition}} \rightarrow$	$\underbrace{(coplay(sr_2, lr_1) \wedge reserved(x, sr_2))}_{\text{(a)}} \wedge$	$\underbrace{(satisfy(x, g) \wedge wanted(g, sr_2))}_{\text{(b)}}$	**Case:**
T	T	F	**3** violation
		$(\neg \mathbf{satisfy}(\mathbf{x}, \mathbf{g}) \wedge wanted(g, sr_2))$	**3a**
		$(satisfy(x, g) \wedge \neg \mathbf{wanted}(\mathbf{g}, \mathbf{sr_2}))$	**3b**
		$(\neg \mathbf{satisfy}(\mathbf{x}, \mathbf{g}) \wedge \neg \mathbf{wanted}(\mathbf{g}, \mathbf{sr_2}))$	**3c**

Table 5.4: Case 3: the reserve schema applies and condition (b) is not met.

[4]Note that in this case the lack of the *coplay* relationship for the `Family Doctor` also implies that the model is laking information regarding which norms are actually applicable for the Social Role.

Case 3 represents the case where the reserve schema is applicable (there is a *reserved* relationship in the legal model) but one of the two condition — condition (b) — is not met. Condition (b) is false in three cases which are summarized in table 5.4: either the *satisfy* relationship is missing, the *wanted* relationship is missing, or both are.

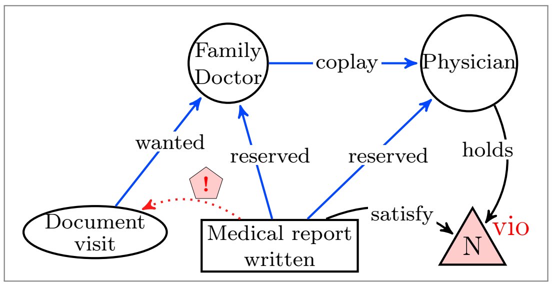


Figure 5.6: Schema violation 3a.

Schema violation 3a. This violation of the schema arises when the *satisfy* relationship is missing in the requirements model ($\neg satisfy(x, g)$ in table 5.3). This case corresponds to the scenario where the restriction is represented in the requirements model — there is the relationship $reserved(x, sr_2)$ — but the information regarding the link between the restriction and the operationalization of the goal is missing. Basically having this relationship makes sure that the Situation is responsibility of the correct Social Role: $satisfy(x, g)$ implies that the Social Role who wants g is responsible for the Situations satisfying g. In the example of figure 5.6 when in the requirements model it is not represented that the Situation "Medical report written" satisfies the Goal "Document visit", then the information regarding the responsibility of the `Family Doctor` w.r.t. the satisfaction of the reserved Situation is not captured.

Mitigation Techniques. For this type of violation, correctly including in the requirements model the missing relationship is sufficient to resolve the problem.

Mitigation: `add:` $satisfy(x, g)$

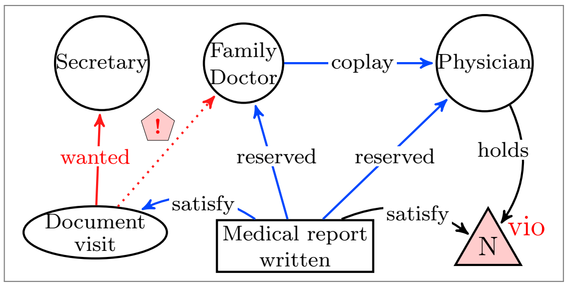


Figure 5.7: Schema violation 3b.

Schema violation 3b. This violation of the schema arises when the *wanted* relationship is missing in the requirements model ($\neg wanted(sr_2, g) \equiv wanted(g, sr_3)$ in table 5.3). By definition, if a Goal exists, then there must be a Social Role who wants it. So if the the *wanted* relationship between the Goal and sr_2 does not exists, then there must be another Social Role sr_3 who wants the Goal.

This case corresponds to the scenario where the the Situation is reserved for the correct Social Role, however it operationalized for the wrong one. In the example of figure 5.7, in the requirements model the Situation "Medical report written" is *reserved* the Social Role who correctly coplays the `Physician`, however the Goal "Document visit" (satisfied by the reserved Situation) is operationalized for the `Secretary`, so the Physician's duty is violated.

Mitigation Techniques. For this type of violation, including in the requirements model the relationship to the right Social Role and removing the one to the wrong Social Role, is sufficient to resolve the problem.

Mitigation 2: `delete:` $wanted(g,sr_3)$;
`add:` $wanted(g,sr_2)$

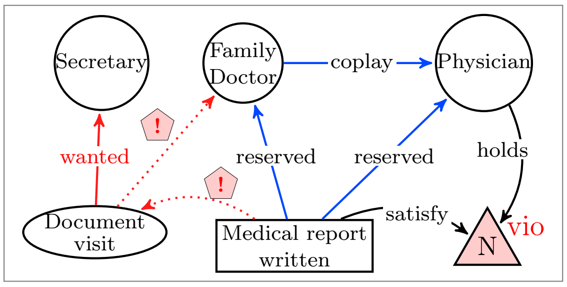


Figure 5.8: Schema violation 3c.

Schema violation 3c. This violation of the schema arises when both the relationships in the (b) conditions are missing $((\neg satisfy(x,g) \wedge \neg wanted(g,sr_2))$. This scenario corresponds to the combination of the two cases before: 2a the Situation is not assigned to the Social Role and 2b the Situation it is operationalized for the wrong Social Role.

In the example of figure 5.8, in the requirements model the Situation "Medical report written" can be assigned to the Social Role only through the two relationships *wanted* and *satisfy*. Since they are both missing, the `Family Doctor` is not responsible for writing the medical report. Moreover — since the Goal "Document visit" (satisfied by the reserved Situation) is operationalized for the `Secretary` — the Physician duty is violated by the fact that the Social Role does not have a Goal operationalizing the reserved Situation.

Case 4: Violation

Reserve Schema: $\underbrace{reserved(x,lr_1)}_{\text{condition}} \rightarrow$	$\underbrace{coplay(sr_2,lr_1) \wedge reserved(x,lr_2)}_{\text{(a)}}$ $\wedge$	$\underbrace{satisfy(x,g) \wedge wanted(g,sr_2)}_{\text{(b)}}$	**Case:**
T	F	F	**4** violation
	$(\neg \mathbf{coplay}(\mathbf{sr_2},\mathbf{lr_1}) \wedge reserved(x,lr_2))$	$(\neg \mathbf{satisfy}(\mathbf{x},\mathbf{g}) \wedge wanted(g,sr_2))$	
	$(coplay(sr_2,lr_1) \wedge \neg \mathbf{reserved}(\mathbf{x},\mathbf{lr_2}))$	$(satisfy(x,g) \wedge \neg \mathbf{wanted}(\mathbf{g},\mathbf{sr_2}))$	
	$(\neg \mathbf{coplay}(\mathbf{sr_2},\mathbf{lr_1}) \wedge \neg \mathbf{reserved}(\mathbf{x},\mathbf{lr_2}))$	$(\neg \mathbf{satisfy}(\mathbf{x},\mathbf{g}) \wedge \neg \mathbf{wanted}(\mathbf{g},\mathbf{sr_2}))$	

Table 5.5: Case 4: the reserve schema applies and neither conditions are met.

Case 4 represents the case where the reserve schema is applicable (there is a *reserved* relationship in the legal model) but both conditions (a) and (b) are not met. These type of violations happen when one of the violations in Case 2 happens jointly with a violation in Case 3 (see also table 5.5):

1. **Case 2a+3a**: $(\neg \mathbf{coplay}(\mathbf{sr_2},\mathbf{lr_1}) \wedge reserved(x,lr_2))$ and $(\neg \mathbf{satisfy}(\mathbf{x},\mathbf{g}) \wedge wanted(g,sr_2))$
2. **Case 2a+3b**: $(\neg \mathbf{coplay}(\mathbf{sr_2},\mathbf{lr_1}) \wedge reserved(x,lr_2))$ and $(satisfy(x,g) \wedge \neg \mathbf{wanted}(\mathbf{g},\mathbf{sr_2}))$
3. **Case 2a+3c**: $(\neg \mathbf{coplay}(\mathbf{sr_2},\mathbf{lr_1}) \wedge reserved(x,lr_2))$ and $(\neg \mathbf{satisfy}(\mathbf{x},\mathbf{g}) \wedge \neg \mathbf{wanted}(\mathbf{g},\mathbf{sr_2}))$
4. **Case 2b+3a**: $(coplay(sr_2,lr_1) \wedge \neg \mathbf{reserved}(\mathbf{x},\mathbf{lr_2}))$ and $(\neg \mathbf{satisfy}(\mathbf{x},\mathbf{g}) \wedge wanted(g,sr_2))$
5. **Case 2b+3b**: $(coplay(sr_2,lr_1) \wedge \neg \mathbf{reserved}(\mathbf{x},\mathbf{lr_2}))$ and $(satisfy(x,g) \wedge \neg \mathbf{wanted}(\mathbf{g},\mathbf{sr_2}))$

6. **Case 2b+3c**: $(coplay(sr_2, lr_1) \wedge \neg\mathbf{reserved}(\mathbf{x}, \mathbf{lr_2}))$ and $(\neg\mathbf{satisfy}(\mathbf{x}, \mathbf{g}) \wedge \neg\mathbf{wanted}(\mathbf{g}, \mathbf{sr_2}))$
7. **Case 2c+3a**: $(\neg\mathbf{coplay}(\mathbf{sr_2}, \mathbf{lr_1}) \wedge \neg\mathbf{reserved}(\mathbf{x}, \mathbf{lr_2}))$ and $(\neg\mathbf{satisfy}(\mathbf{x}, \mathbf{g}) \wedge wanted(g, sr_2))$
8. **Case 2c+3b**: $(\neg\mathbf{coplay}(\mathbf{sr_2}, \mathbf{lr_1}) \wedge \neg\mathbf{reserved}(\mathbf{x}, \mathbf{lr_2}))$ and $(satisfy(x, g) \wedge \neg\mathbf{wanted}(\mathbf{g}, \mathbf{sr_2}))$
9. **Case 2c+3c**: $(\neg\mathbf{coplay}(\mathbf{sr_2}, \mathbf{lr_1}) \wedge \neg\mathbf{reserved}(\mathbf{x}, \mathbf{lr_2}))$ and $(\neg\mathbf{satisfy}(\mathbf{x}, \mathbf{g}) \wedge \neg\mathbf{wanted}(\mathbf{g}, \mathbf{sr_2}))$

The two simultaneous violations should be dealt separately in order to resolve the violation.

For example when **Case 2b+3a** occur (number 4 above), we have a case where: in condition (a) the *reserved* relationship is missing (Case 2b), and in condition (b) the *satisfy* relationship is missing. The two mitigation technique for the two violations should then be joint:
`add` $reserved(x, sr_2)$ relation;
`add` $satisfy(x, g)$ relation)

Case 5: No violation

Reserve Schema:			
$\underbrace{reserved(x, lr_1)}_{\text{condition}} \rightarrow$	$\underbrace{coplay(sr_2, lr_1) \wedge reserved(x, lr_2)}_{\text{(a)}} \wedge$	$\underbrace{satisfy(x, g) \wedge wanted(g, sr_2)}_{\text{(b)}}$	**Case:**
F	*	*	**5** no violation

Table 5.6: Case 5: the reserve schema does not apply.

Case 5 represents the case where the reserve schema is not applicable: the model of the law does not contain the *reserved* relation, so it is not necessary that the requirements model either condition (a) or (b) is met (table 5.6). Basically when the law does not specify that a given responsibility is assigned to only one Role, then it is not required that this restriction is matched in the requirements model. So in this case the analyst can still decide to model this restriction and assign a Situation only to a specific Social Role, however it is not relevant from the point of view of compliance.

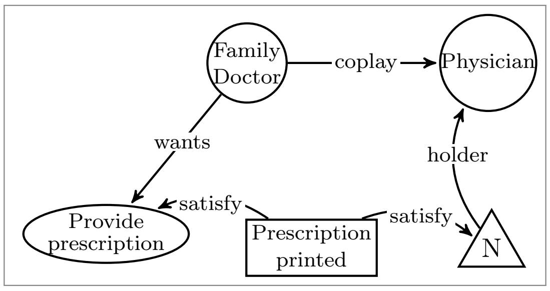


Figure 5.9: Schema not applicable.

Figure 5.9 shows an example where a Norm requires that the `Physician` provides the patient with the prescription. The law does not mandate that *only* the `Physician` can perform such action, so the legal model does not have the *reserved* relation. So in the requirements model as long as the prescription is printed and given to the patient (i.e., as long as the Situations are satisfied), the Norm is complied with.

5.1.3 Legally-valid social delegation

The objective of Nòmos 3 is to be able to support the design of a system that both meets its purpose and respects the applicable norms. In most GORE approaches (e.g., i* Yu [1997a]) actors/roles can depend on others to achieve their objective. The concepts of dependency — more formally delegation in Giorgini et al. [2005a] — is therefore an important design strategy to be considered in our models.

However, when a responsibility is reserved to a specific role, delegating the responsibility in the requirements model must occur in a way that it does not introduces a violation in the legal model. For example, a Physician may want to delegate some of his/her responsibilities to a Secretary, Nurse or another Physician, but not all responsibilities can be delegated to all roles in the system. Intuitively, a Physician should not be allowed to delegate the Secretary to "Prescribe medicine for patient" but delegating it to another Physician would be fine. Basically in presence of a delegation it is necessary to check for cases of "right-action-wrong-role" in order to avoid violations.

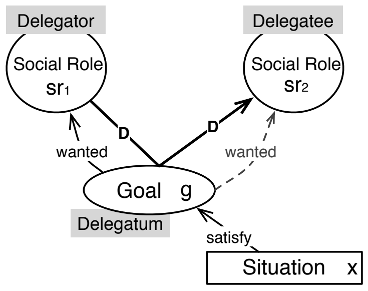


Figure 5.10: Example of Social Delegation in Nòmos 3.

In Nòmos 3, a social delegation is modelled as a ternary relationship among two Social Roles (the delegator and the delegatee) and a Goal (the delegatum). In figure 5.10 we have represented the delegation relationship ($\xrightarrow{D}$) between the two Social Roles sr_1 and sr_2, and over Goal g. The Goal is satisfied by Situation x which is assigned to the first Social Role. The delegation corresponds to the delegatee also wanting the same Goal (dotted relationship $g \xrightarrow{\text{wanted}} sr_2$) and begin responsible for the same Situation x. The delegatee may however not be entitled to bring about the Situation, and the reserve schema identifies these cases.

The consideration that some social delegation in the requirements model may make the system in violation with some Norms, led us to define the concept of legally-valid social delegation.[5]

Def. 1 *A* **legally-valid social delegation** *is the delegation of a Goal in the requirements model, which does not cause a violation of the reserve schema.*

A social delegation is legally-valid if:

(a) it involves a Goal satisfied by a Situation not reserved by law to any specific Legal Role, or

(b) it concerns a Goal reserved for a Social Role and the delegated Social Role is entitled to receive it.

[5]We use the term *social* delegation as we are dealing with a social artifacts and not a delegation from the legal point of view.

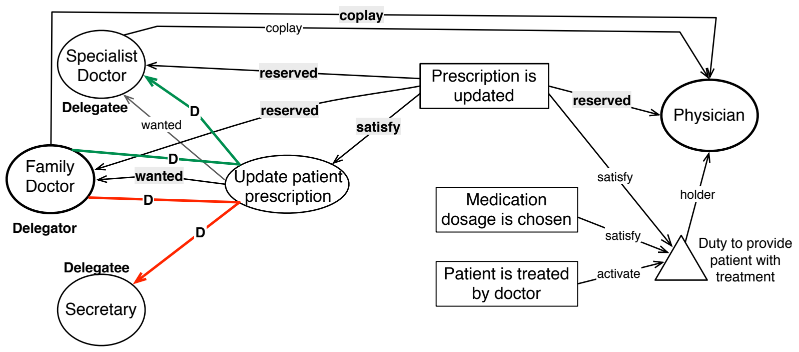


Figure 5.11: Example of two social delegation: only one is legally valid.

On the contrary, a social delegation is *not* legally-valid if it transfers a Goal from a Social Role for which it is reserved, to another Social Role not entitled to receive it.

For example, in figure 5.11 the Social Role of `Specialist Doctor` also coplays the Legal Role of `Physician`. So if the analyst decides to allow the `Family Doctor` to delegate the Goal of "Update patient prescription" to the `Specialist Doctor`, the reserve schema applies and is respected: the Situation "Prescription is updated" is also *reserved* for the `Specialist Doctor` who also coplays the Legal Role of `Physician`. If instead the analyst decides to allow the `Family Doctor` to delegate the Goal "Update patient prescription" to the `Secretary` (relationship in red in figure 5.11), this delegation makes the `Secretary` responsible for the Goal and for bringing about the Situation satisfying the Goal. The `Secretary` is however not allowed to bring about the Situation "Prescription is updated" which is instead reserved for the Social Role coplaying the `Physician`. So including this last delegation in the model corresponds to the Schema Violation 3 for the duty of the `Physician` to provide the patient with treatment.

5.2 Goal model compliance

In Nòmos 3 a norm's compliance depends on three factors: (i) the situations that make a norm applicable, (ii) the situations that actually do what the norm says (make the norm satisfied), and lastly (iii) the roles that bring about such situations. In the previous section we have introduced a new mechanism based on a schema of Nòmos 3 relations, used to identify cases in which a Norm is violated because a Situation is not assigned to the right Role.

In this section we expand the reasoning over the compliance of a Nòmos 3 model — introduced in section 4.1 with the compliance of a Nòmos 3 legal model — to include also the violation that arise from the reserve schema. A combination of forward/backward reasoning algorithms together with the identification of violations of the reserve schema, are applied to Nòmos 3 models to answer two questions regarding the compliance of a set of Goals:

1. Given a set of Goals in a Nòmos 3 model, which compliance value have the Norms?

2. Which Situations should be brought about in a Nòmos 3 model to achieve the desired compliance value of a set of Norms?
   And which mitigation techniques should be implemented in the Nòmos 3 model to achieve the desired compliance value?

In the following we introduce the different algorithms that allow the analyst the modeller to answer the questions above. The running scenario that will be used to illustrate the reasoning algorithms is the same as in chapter 4.

5.2.1 Goal model compliance analysis

Full compliance analysis is an algorithm intended to find the compliance value of a set of Norms given a set of Goals represented by a set of Situations with an initial satisfaction value. This algorithm is used to answer question 1 above: Given a set of Goals in a Nòmos 3 model, which compliance value have the Norms?

The input to our algorithm is the Nòmos 3 model M and a list of Situations with their satisfaction value ($IntialSit[sit\_id, sat\_val]$). The output of the algorithm is a list describing the compliance value of each Norm $solution[norm\_id, compl\_val]$. The algorithm first uses the simple-compliance-analysis algorithm (line 2), the preliminary solution is identified with the compliance value that the norm obtain from the value-propagated algorithm. Then through a set of four cycles, the algorithm scans for schema violations in order to update the compliance value of norms in case of violation. For every *reserved* relationship in the legal model, it identifies the Norm n_i satisfied by the reserved Situation (line 2). First the algorithm finds the Social Role coplaying that Legal Role (line 3) and if it does not find the *coplay* relation, it identifies a Schema violation 2a and adds to the solution the schema violation `vio-2a` for the Norm n_i (line 4-5). Then it finds the *reserved* relationship from the Situation to the Social Role (line 7) and if it does not find it, it identifies a Schema violation 2b and adds to the solution the schema violation `vio-2b` for the Norm n_i (line 8-9). The algorithm continues and tries to identify the $satisfy$ relationship from the Situation to the Goal (line 11) and if it does not find it, it identifies a Schema violation 3a and adds to the solution the schema violation `vio-3a` for the Norm n_i (line 12-13). Lastly, the algorithm tries to identify the *wanted* relationship that makes the wrong Social Role (sr_j) responsible for the Goal and Situation assigned to the other Social Role sr_i (line 16-17). Since these four steps are sequential, an occurrence of multiple schema violations happening jointly, is recorded in the multiple violations value for the Norm. The algorithm then returns the solution array with the list of Norms and compliance values.

Example. In the running scenario of the system for managing patients' health in an hospital, we will consider in input to the algorithm the Nòmos 3 model in figure 5.12, and the 7 Situations value describing the following scenario. The Family Doctor updates the previously-chosen prescription of a Patient, and the dosage has been chosen as well ($(s_1, s_2, s_3) = \mathrm{ST}$). The Secretary informs

```
**Input**: $M$, $IntialSit[sit\_id, sat\_val]$
**Output**: $solution[norm\_id, compl\_val]$
1  solution[] = simple-compl-analysis($M$,$IntialSit$);
2  foreach $reserved(x, lr_i) \in M$ such that $satisfy(x, n_i)$ do
3      Find $sr_i \in M$ such that $coplay(sr_i, lr_i)$
4          else /* Schema violation 2a /*
5              solution.add($n_i$, vio-2a);
6      end
7      Find $reserved(x, sr_i) \in M$
8          else /* Schema violation 2b /*
9              solution.add($n_i$, vio-2b);
10     end
11     Find $g \in M$ such that $satisfy(x, g)$
12         else /* Schema violation 3a /*
13             solution.add($n_i$, vio-3a);
14     end
15     Find $wanted(g, sr_j)$ such that $sr_j \neq sr_i$
16         /* Schema violation 3b /*
17         solution.add($n_i$, vio-3b);
18     end
19     return solution;
20 end
```

Algorithm 11: Full compliance analysis (full-compl-analysis()). This algorithm evaluates the compliance value of a set of Norms, with respect to an initial set of Goals (represented by a set of Situations holding $IntialSit[sit\_id, sat\_val]$) using first the simple compliance-analysis() algorithm, and then evaluating possible violations of the reserve schema in the input model (M).

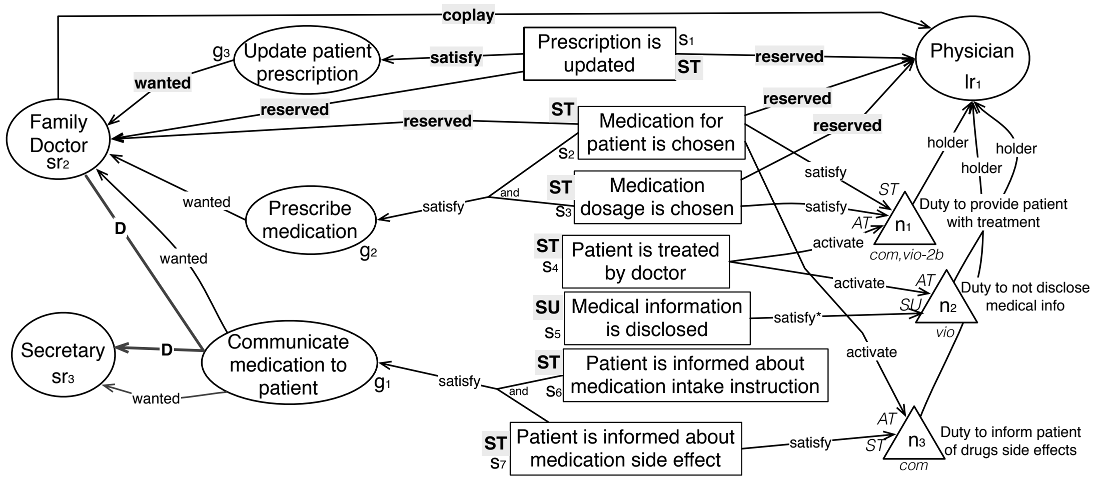

coplay
g3 Update patient prescription
satisfy
Prescription is updated
s1 ST
reserved
Physician lr1
wanted
reserved
reserved
Family Doctor sr2
ST Medication for patient is chosen s2
reserved
reserved
holder
holder
holder
wanted
Prescribe medication g2
satisfy
and
ST Medication s3 dosage is chosen
satisfy
satisfy
ST
AT
n1
Duty to provide patient with treatment
com,vio-2b
D
wanted
ST s4 Patient is treated by doctor
activate
activate
AT
SU s5 Medical information is disclosed
satisfy*
SU
n2
Duty to not disclose medical info
vio
Secretary sr3
D
Communicate medication to patient
g1
wanted
satisfy
and
ST s6 Patient is informed about medication intake instruction
activate
ST s7 Patient is informed about medication side effect
satisfy
AT
ST
n3
Duty to inform patient of drugs side effects
com

Figure 5.12: Full compliance analysis: scenario of the online seller.

the Patient about intake instructions and side effect ($sat(s_6, s_7) = \text{ST}$). There is no information on whether medical information has been disclosed ($sat(s_5) = \text{SU}$).

First the algorithm initiates the solution set by using the simple-compl-analysis algorithm. The first norm n_1 is evaluated as compliant *com*: the patient is treated by a doctor and both situations satisfying the norm are satisfied $((s_2, s_3) = \mathrm{ST})$. Since there is no evidence about information disclosure, the second norm n_2 is applicable and has undefined satisfaction, which makes the norm violated *vio*. The last norm n_3 is both applicable and satisfied $((s_2, s_7) = \mathrm{ST})$, making the norm compliant *com*. Then, the algorithm searches for *reserved* relationships in the legal model and identifies whether the reserve schema has been violated. Three relationships are identified:

1. $s_1 \xrightarrow{\text{reserved}} lr_1$ For this relationship the algorithm searches first for the *coplay* relationship linking the Legal to the Social Role $(sr_2 \xrightarrow{\text{coplay}} lr_1)$. Then it searches for the *reserved* relationship in the requirements model $(s_1 \xrightarrow{\text{reserved}} sr_2)$, and for the relationships corresponding to the correct operationalization of the goal $(s_1 \xrightarrow{\text{satisfy}} g_3$, and $g_3 \xrightarrow{\text{wanted}} sr_2)$. No schema violations is identified for this relation.

2. $s_2 \xrightarrow{\text{reserved}} lr_1$ For this relationship the algorithm searches first for the *coplay* relationship $(sr_2 \xrightarrow{\text{coplay}} lr_1)$. Then it searches for the *reserved* relationship in the requirements model $(s_2 \xrightarrow{\text{reserved}} sr_2)$. Lastly it looks for the relationships corresponding to the correct operationalization of the goal $(s_2 \xrightarrow{\text{satisfy}} g_2$, and $g_2 \xrightarrow{\text{wanted}} sr_2)$. No schema violations is identified for this relation.

3. $s_3 \xrightarrow{\text{reserved}} lr_1$ For this last relationship the algorithm first finds the *coplay* relationship $(sr_3 \xrightarrow{\text{coplay}} lr_1)$. Then it searches for the *reserved* relationship in the requirements model and does not find it: this missing relationship triggers a schema violation 2a, and the compliance value of the norm n_1 is updated to $vio - 2b$. Lastly the algorithm searches for the relationships corresponding to the correct operationalization of the goal $(s_3 \xrightarrow{\text{satisfy}} g_2$, and $g_2 \xrightarrow{\text{wanted}} sr_2)$. No schema violations is identified for this relation.

The output of the algorithm then shows that two norms are violated (n_1 and n_2), while only n_3 is compliant.

5.2.2 Goal model compliance search

Full compliance search is an algorithm intended to find the satisfaction value that a set of Goals in a Nòmos 3 model must have, to obtain the desired compliance of a set of Norms. Since compliance may depend on the relationships in the model, the algorithms also identifies the changes to the model (expressed as mitigation techniques) that needs to be brought about to achieve such compliance. This algorithm is used to answer question 2 above: Which Goals should be brought about in a Nòmos 3 model to achieve the desired compliance value of a set of Norms? And which mitigation techniques should be implemented in the Nòmos 3 model to achieve the desired compliance value of the set of Norms?

The input to our algorithm is the Nòmos 3 model M and a list of Norms with the associated desired compliance value ($DesiredVal[norm\_id, comp\_val]$). The output of the algorithm are two lists: one describing the satisfiability value for the Situations in the model $solution[sit\_id, sat\_val]$, the other the list of changes to implement in the model for the compliance to be achieved ($mitigationRules[]$). The algorithm consists in an application of a backward reasoning algorithm that searches for assignments of values to Situations which would propagate the desired values to the target nodes. At the beginning the boolean variable indicating that a solution has been found is set to false (line 1). Then the algorithm generates an assignment to the Situations in the model (line 4), and uses the full-compl-analysis() algorithm to evaluate the compliance value of the Norms using this assignment (line 5). Before deciding whether a solution has been found, the algorithm updates the compliance value of the norms that have a violation deriving from the reserve schema. When a Norm has a value that identifies a Schema Violation 2a, the algorithms adds the mitigation rule in the list of changes to implement in the model, and then updates the value from `vio-2a` to `com` (line 6-9)—indeed implementing the mitigation rule would result in the violation being resolved. The same procedure is performed to resolve Schema Violations 2b (line 10-13), Schema Violations 3a (line 14-17), Schema Violations 3b (line 18-22). Then if all Norms in the model have the desired compliance value (line 23), then a solution has been found: the boolean variable is updated (line 25) and the solution and mitigation changes are returned (line 28-31). Otherwise if not all the Norms have the desired compliance value, another assignment is generated and the process is repeated: this loop is repeated until either a solution is found or all the space of alternatives has been explored (line 27), in which case an empty solution set is returned (line 33).

```
**Input**: $M$, $DesiredVal[norm\_id, comp\_val]$
**Output**: $solution[sit\_id, sat\_val], mitigationRules[]$
boolean sol = false;
mitigationRules[];
repeat
    generate $assignment_i[sit\_id, sat\_val]$;
    $tmp\_sol[norm\_id, compl\_val]$ = full-compl-analysis($M, assignment_i$);
    /* for norm violated by the schema, add mitigation rules, update norm status */
    foreach $n_i$ in $tmp\_sol[]$ such that: $n_i.compl\_val()$=vio-2a do
        mitigationRules.add("add: $coplay(sr_i, lr_i)$")
        tmp_sol.update($n_i$, com);
    end
    foreach $n_i$ in $tmp\_sol[]$ such that: $n_i.compl\_val()$=vio-2b do
        mitigationRules.add("add: $reserved(x, sr_i)$")
        tmp_sol.update($n_i$, com);
    end
    foreach $n_i$ in $tmp\_sol[]$ such that: $n_i.compl\_val()$=vio-3a do
        mitigationRules.add("add: $satisfy(x, g)$")
        tmp_sol.update($n_i$, com);
    end
    foreach $n_i$ in $tmp\_sol[]$ such that: $n_i.compl\_val()$=vio-3b do
        mitigationRules.add("delete: $wanted(g, sr_j)$")
        mitigationRules.add("add: $wanted(g, sr_i)$")
        tmp_sol.update($n_i$, com);
    end
    /* when all norms have the desired compliance value, a solution has been found */
    if $\forall n_i$ in $M$: $tmp\_sol.n_i.comp\_value() == DesiredVal.n_i.comp\_value()$;
    then
        sol=true
    end
until sol=true or space alternatives explored;
if sol=true then
    $solution[] = tmp\_sol.r_i[]$;
    return $solution[]$;
    return $mitigationRules[]$;
else
    return $solution[]$
end
```

Algorithm 12: Full compliance search (full-compl-search()). This algorithm evaluates the satisfaction value that a set of Situations in a Nòmos 3 model (M) must have to achieve the desired compliance value of a set of Norms ($DesiredVal[norm\_id, comp\_val]$). Moreover it returns the list of changes that should be applied to the model in order to resolve the violations arising from the reserve schema ($mitigationRules[]$).

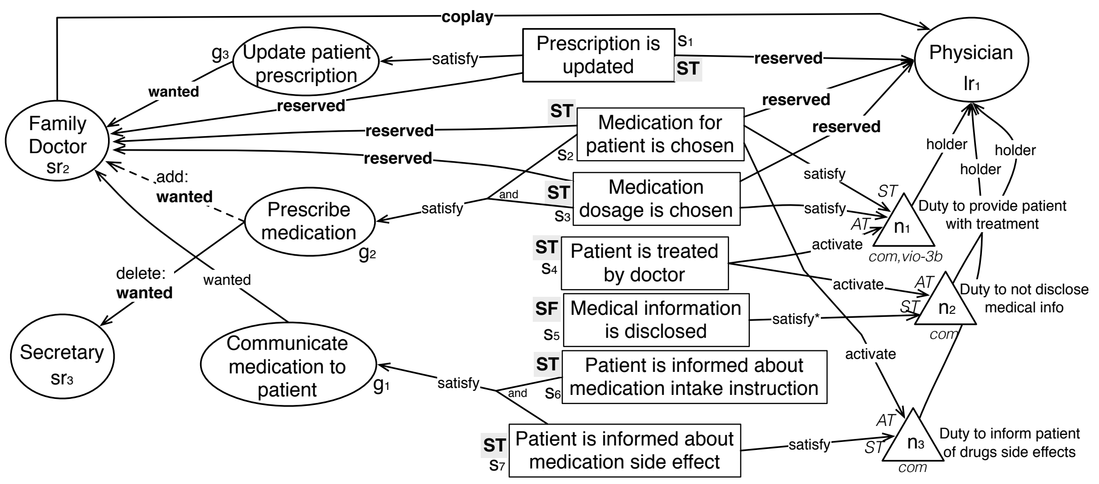

coplay
Physician
lr1
g3
Update patient prescription
satisfy
Prescription is updated
s1
ST
reserved
wanted
reserved
Family Doctor
sr2
ST
Medication for patient is chosen
s2
reserved
reserved
reserved
reserved
add:
wanted
Prescribe medication
g2
satisfy
and
ST
Medication dosage is chosen
s3
satisfy
satisfy
holder
holder
holder
ST
AT
n1
Duty to provide patient with treatment
com,vio-3b
ST
s4
Patient is treated by doctor
activate
activate
delete:
wanted
wanted
Secretary
sr3
Communicate medication to patient
g1
SF
s5
Medical information is disclosed
satisfy*
AT
ST
n2
Duty to not disclose medical info
com
activate
satisfy
and
ST
s6
Patient is informed about medication intake instruction
ST
s7
Patient is informed about medication side effect
satisfy
AT
ST
n3
Duty to inform patient of drugs side effects
com

Figure 5.13: Full compliance search: scenario of the online seller.

Example. In the running scenario of the system for managing patients' health in an hospital, we will consider in input to the algorithm the Nòmos 3 model in figure 5.13, and the desired compliance value for the 3 norms is compliant *com*.

After initiating the boolean variable to false (a solution has not yet been found) and initiating to empty the set of mitigation rules, the algorithm repeats the loop (line 3–27) until a solution has been found or all alternatives have been explored without success. In the following we will consider the assignment represented in figure 5.13, where all situations are satisfied ST, except for s_5 which is not satisfied ($sat(s_5)$ = SF). The algorithm uses this assignment and feeds it to the full-compl-analysis() algorithm to evaluate the compliance value of the norms. Since s_4 and s_2 are holding, all three norms are applicable. s_2, s_3, s_7 are also satisfied, and make n_1 and n_3 satisfied. The norm n_2 is a duty not to disclose medical info so when s_5 "Medical info is disclosed" is false, the $satisfy^*$ relationship allows the modeler to identify that the norm has been correctly satisfied (as no information is disclosed). However when the full-compl-analysis() algorithm evaluates the *reserved* relationships in the legal model and the possible violation of the schema, it is identified that goal g_2 is wanted by the wrong role (sr_3 does not coplay the legal role) and the compliance value `vio-3b` is added to the norm n_1. The full-compl-search() algorithm then evaluates the compliance value returned by the full compliance analysis, and adds to the solution the mitigations rules necessary to resolve violation resulting from the reserve schema (line 6-22). In the current assignment we have that n_1 has values $com, vio - 3b$, n_2 and n_3 have value *com*. Since only one norm has a compliance value resulting from a violation of the reserve schema, the algorithm skips the first 3 foreach loops (none of the norms have a compliance value of $vio - 2a$,$vio - 2b$ or $vio - 3a$) and executes the last corresponding to a schema violation 3b. Two operations are added to the mitigation rules: first the wrong *wanted* relationship from g_2 to the Secretary sr_3 should be deleted, and secondly the correct *wanted* relationship from g_2 to the Family Doctor sr_2 should be added. In light of these two changes which would resolve the schema violation, the compliance value of the norm is updated to compliant *com*. Since all three norms have now the desired compliance value, the assignment to the situations is returned as a solution together with the necessary mitigation rules that needs to be implemented in the model.

5.3 Conclusions

The objective of this chapter was to illustrate the reasoning that allows the modeler/analyst to reason over all three dimensions around the compliance of a norm: not only the Situations defining a Norm's applicability and satisfiability, but also the Role who brings about such Situations. While in the previous chapter the reasoning was focused only on the satisfiability of the Situations and how bringing about different situations influenced the compliance of a Norm, in this chapter the focus was on being able to identify cases where a role brings about the right situations but since the role is not entitled to do so, a norms violation is detected.

In this chapter we have introduced an important new mechanism to detect the violation of a norm when the the wrong role — the one not entitle to do so — brings about the Situations satisfying a Norm. In order to identify these cases of "right-action-wrong-role", and support the corresponding analysis in the models, we have focused on the only relationship in the Nòmos 3 model that indeed embodies the restriction where only a given Role can bring about a Situation: the *reserved* relation. In order to identify violations we use a schema of relationships in the Nòmos 3 model, called the reserve schema. The core idea behind the schema is to identify cases of "right-action-wrong-role": if in the law it is specified that a given responsibility is assigned only to a specific Legal Role, then in the requirements model we must check that the corresponding Social Role is also the only responsible. Moreover, when a responsibility is reserved to a specific role, delegating the responsibility in the requirements model must occur in a way that it does not introduces a violation in the legal model. So since a social delegation in the requirements model may make the system in violation with some Norms, we defined the concept of *legally-valid social delegation* as the delegation of a Goal from a Social Role to another in the requirements model, which does not cause a violation of the reserve schema. Lastly in the chapter we have expanded the reasoning over the compliance of a Nòmos 3 model: a combination of forward/backward reasoning algorithms together with the identification of violations of the reserve schema, are applied to Nòmos 3 models to answer questions regarding the compliance of a set of Norms.

Chapter 6

Systematic process for revising requirements

In the previous chapters we have presented the Nòmos 3 modeling language and the different reasoning algorithms that allow the analyst to answer several important questions about the model. In order to help the analyst manage the revision of a requirements model to achieve its compliance with some norms, it is important to structure and guide the analyst through the different steps needed to revise the model while using the proposed algorithms.

The objective of this chapter is therefore to describe our proposed systematic process for helping the analyst establish the compliance of a set of requirements with a given piece of law in a Nòmos 3 model. The chapter will first present an overview of the process and its three phases — analysis, evaluation, and revision — and will then present all the steps in details.

6.1 Overview of the process

In this section we provide an overview of the systematic process for establishing compliance of a set of requirements. The goal of this process is to guide the analyst through the analysis and revision a Nòmos 3 requirements model using the reasoning capabilities offered by our proposed language (see chapter 4 and 5).

The input to our procedure is a Nòmos 3 model of the requirements, and Nòmos 3 model of the law. Our approach outlines all the steps that the analyst need to do to manage the compliance of the modeled requirements with the included model of the law.

The process of making an input set of requirements compliant with a given model of law, is a procedure that involves three phases:

1. The analysis of the model is the first phase of our process. The objective of this phase is first to allow the analyst to perform an informative evaluation regarding which norms apply and which social and legal roles are involved (or missing). In this evaluation the

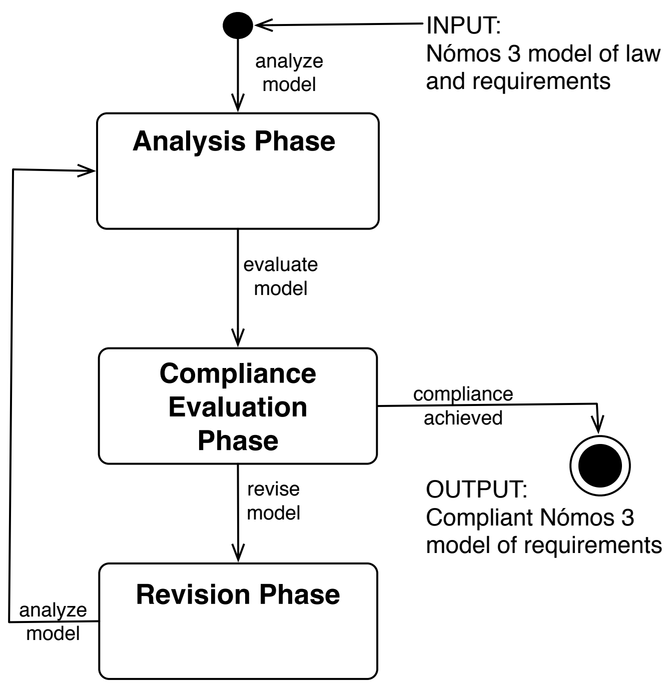


Figure 6.1: Workflow of the proposed compliance process

analyst will have the possibility to perform some small revision of the model according to the result of the applicability analysis performed. Once the model has been analyzed it needs to be evaluated for compliance.

2. In the second phase the actual compliance evaluation is performed. The objective of this phase is to evaluate whether the current model of requirements meets the desired compliance target for the norms. If compliance is achieved the Nòmos 3 model is returned in output, otherwise the model is passed to the next phase that involves a revision of the model.

3. The last phase is indeed the revision of the model. The objective of this phase is to take as input a non-compliant model and apply different strategies to help the analyst solve these problems. The model produced is then passed on to the first analysis phase before it is evaluated again for compliance.

In figure 6.1 we have summarized the three phases of our compliance process. The input to the process is a Nòmos 3 model of law and requirements, and the first activity is to analyze the model in the analysis phase. The model is then evaluated in the compliance evaluation phase: if compliance is achieved the model is returned in output, otherwise the model is revised in the revision phase. From this last phase, the process further analyze the model in the analysis phase before its compliance is evaluated again.

6.1.1 Compliance target

One important characteristic of our proposal is the possibility to taylor and customize the overall compliance of the requirements. As we saw in chapter 3, in Nòmos 3 a norm can have four compliance values:

1. compliance *com*: the norm is applicable and satisfied
2. tolerated *tol*: the norm (a right) is applicable but can be not satisfied, or the norm is not applicable
3. inconclusive *inc*: it is not known if the norm
4. or violated *vio*: the norm is ab applicable duty that is not satisfied, or the norm is involved in a schema violation.

In our procedure we allow the analyst to choose the desired compliance value for norms, however when making such customization there are some considerations to make w.r.t. what it means to chose a specific value.

- Setting a desired compliance value of *com* or *tol* for a norm, implies willingness in complying, and therefore in our process we consider these as default values for achieving compliance.

- Setting a desired compliance value of *inc* for a norm, would imply willingness in not knowing whether the norm applies. If the analyst chooses to set as desired value *inc*, we interpret this as a sign of tolerance in case the norm is evaluated as inconclusive, and we also consider that if the norm is evaluated complied or tolerated, it is also fine. Choosing *inc* as desired compliance value means however that a violation of the norm is not admitted (which would go beyond the *not knowing*).

- Similarly, setting a desired value of *vio* for a norm, should be interpreted as a sign of tolerance wrt to the norm's violation. If the analyst chooses to set as desired value *vio*, we consider that it is acceptable if the norm is evaluated as violated, complied, or tolerated. It is not admitted the case when the norm is inconclusive.

	Target compliance value		
Obtained compliance value	*com*/*tol*	*inc*	*vio*
com/*tol*	✓ok	✓ok	✓ok
inc	✗no	✓ok	✗no
vio	✗no	✗no	✓ok

Table 6.1: Possible combinations of compliance value: target values vs values obtained in the analysis.

In table 6.1 we have summarized these cases about the target and obtained compliance value for a norm, and we have classified them as 'ok' or 'no'. "✓ok" means that either the desired compliance value is obtained in the model, or a better compliance value is achieved (*tol*/*com*). "✗no" means that the desired compliance value not obtained in the model. The classification of these cases reflects the consideration made before regarding the meaning of a desired compliance value.

6.2 The compliance process

In this section we detail all the activities and steps of each phase and provide a more thorough description of the three phases of the compliance process.

6.2.1 Analysis of the model

The objective of this phase is to guide the analyst into a preliminary analysis of the model. This analysis consists in identifying the applicable norms (in general), and the specific norms applicable to the roles in the model. The aim of this analysis is to allow the analyst assess the normative context of the set of requirements at hand, and — if needed — make some small adjustment to the model.

In figure 6.2 we have represented in details the activities of this phase. This phase receives in input a Nòmos 3 model that needs to be evaluated for compliance. Before this evaluation takes place, the analyst is first guided into an analysis of the model.

- First, the analyst uses the reasoning algorithm `applicability-analysis()` (see chapter 4) to perform an initial **analysis of applicable norms**. The evaluation of which Norms are applicable is done by evaluating the Situations brought about in the requirements model and using a forward-propagating algorithm to identify the applicable Norms.
- If the analyst is not satisfied from either analysis, he is guided to perform a **small revision** of the model. Through the backward-propagating reasoning algorithm `applicability-search()` the analyst can study and explore the applicability of the desired norms and can therefore revise the model accordingly: adding or revising goals/domain-assumptions are some of the options that the analyst has to revise the norms applicable to the requirements. After the model has been revised, another analysis of the applicable norm can be performed, or an analysis of roles.
- When the analyst is satisfied and finds that the applicable norms are reasonable, he is guided to an **analysis of roles** and the norms that apply to them. Similarly, if he/she is not satisfied with the norms applicable to a given role, then he is guided back to the small revision of the model where for example he can add missing social roles or revise the model to avoid the introduction of new roles. If also this analysis is satisfactory, the analysis phase terminates.

The output of this phase is the final Nòmos 3 model that needs to be evaluated for compliance.

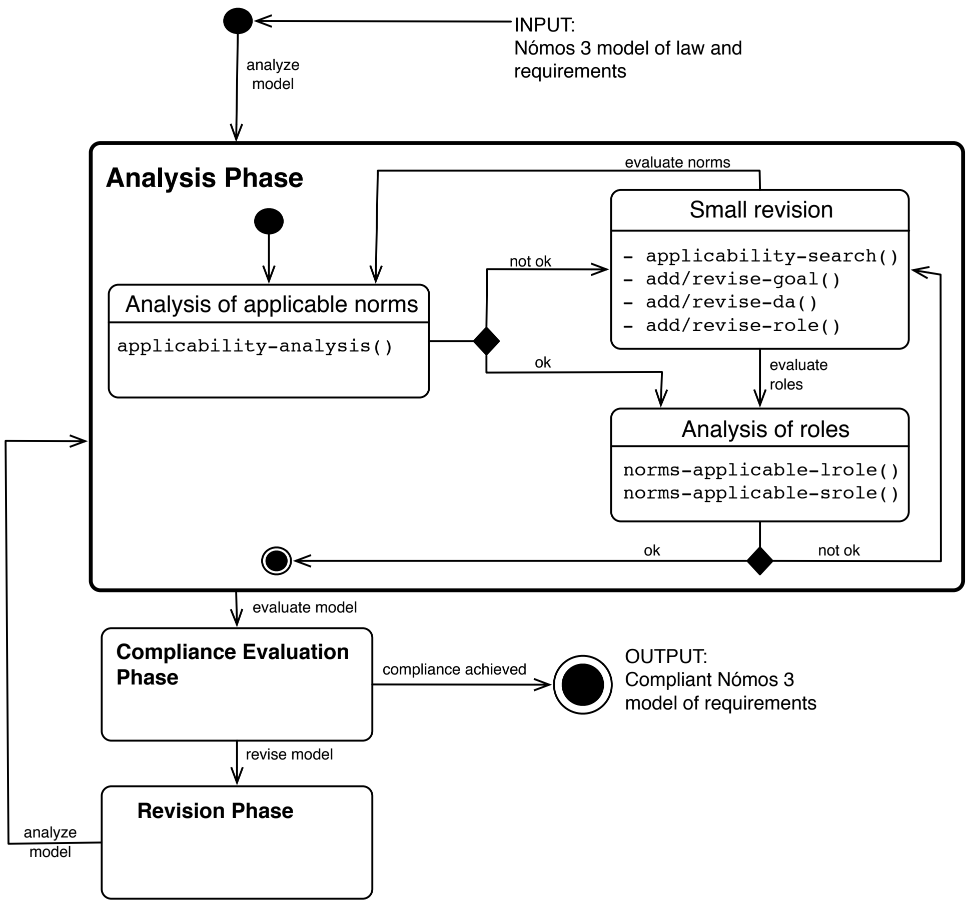


Figure 6.2: Workflow detailing the activities in the analysis phase of the compliance process.

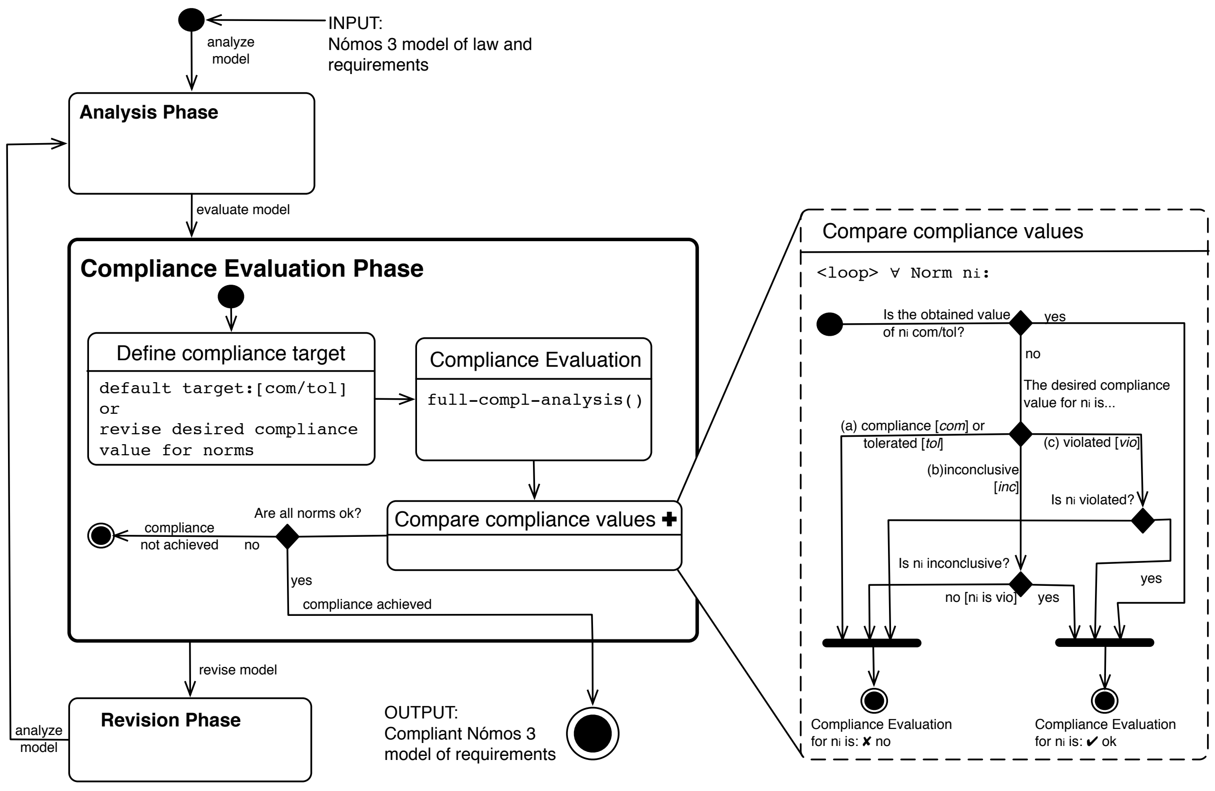

INPUT:
Nómos 3 model of law and requirements
analyze model
Analysis Phase
evaluate model
Compliance Evaluation Phase
Define compliance target
default target:[com/tol] or revise desired compliance value for norms
Compliance Evaluation
full-compl-analysis()
Compare compliance values
Are all norms ok?
compliance not achieved
no
yes
compliance achieved
revise model
Revision Phase
analyze model
OUTPUT:
Compliant Nómos 3 model of requirements
Compare compliance values
<loop> ∀ Norm ni:
Is the obtained value of ni com/tol?
yes
no
The desired compliance value for ni is...
(a) compliance [com] or tolerated [tol]
(b)inconclusive [inc]
(c) violated [vio]
Is ni violated?
Is ni inconclusive?
no [ni is vio]
yes
yes
Compliance Evaluation for ni is: ✘ no
Compliance Evaluation for ni is: ✔ ok

Figure 6.3: Workflow detailing the activities in the compliance check of the compliance process.

6.2.2 Compliance evaluation

The objective of this phase is to perform an evaluation of the compliance of the norm, given a set of requirements. This evaluation consists of three steps. The definition of a compliance target is the first activity, then the actual compliance value of the norms in the model is calculated, and eventually the two compliance values are compared to evaluate if the compliance of the model corresponds to the desired one. The aim of this 3-steps evaluation is indeed to allow

In figure 6.3 we have represented in details the activities of this phase. This phase receives in input the Nòmos 3 model that needs to be evaluated for compliance.

- First the **compliance target is defined**. In this activity the analyst has the possibility to revise and specify the desired compliance value for some/all norms. By default, all norms have a compliance target value of *com*/*tol*.
- The second activity corresponds basically to the execution of the compliance algorithm `full-compl-analysis()` which performs a **compliance evaluation** of the model and returns the compliance value that the norm in the model have, given the set of requirements (see section 5.2).
- Lastly, an automatic **comparison of compliance values** is performed. The flow of this

evaluation is detailed on the side of figure 6.3. This activity basically operationalizes the comparison of compliance values according to table 6.1.

- **Compare compliance values.** The compliance values of every norm is evaluated: the desired compliance value (desired value) vs. the compliance value obtained in the model analysis (obtained value). If the obtained value for a norm n_i is com/tol, then the flow classifies the norm as ok: either the desired value is met — in case it was com/tol — otherwise — in case it was *inc* or *vio* — a positive compliance result is still considered ok.
  If the obtained value is not com/tol, then the desired value is considered. (a) if the desired value was com/tol then for sure the compliance evaluation fails: the obtained value is necessarily either *inc* or *vio*. (b) If the desired value was *inc*, then it is checked if the obtained value is *inc*: if yes, the compliance evaluation is ok, otherwise it is not ok: the obtained value is necessarily *vio*. (c) If the desired value was *vio*, then it is checked if the obtained value is *vio*: if yes, the compliance evaluation is ok, otherwise it is not ok: the obtained value is necessarily *inc*. Each evaluation reaches an end when one of the two compliance value is obtained, and the entire loop terminates once this evaluation has been performed for all norms in the model.

When all the norms in the model have been evaluated, the last check is performed: if the compliance evaluation of all norms is ok, then the model has met the desired compliance target and it is returned in output. Otherwise, if not all norms are ok, then the desired compliance is not achieved and the model is passed to the next phase.

The output of this phase is either a compliant Nòmos 3 model — in which case the process terminates — or a non-compliant Nòmos 3 model that needs to be revised.

6.2.3 Model revision

The objective of this phase is to guide the analyst through the revision of a non-compliant model. This phase consists basically of two activities that are repeated for every norm that was found not-compliant: first the revision strategy is chosen, then it is applied. In these two activities the analyst is guided through a series of questions that will help him/her identify how to fix the model together with the help of the Nòmos 3 reasoning algorithms presented in chapter 4 and 5.

In figure 6.4 we have represented in details the activities of this phase. This phase receives in input a non-compliant Nòmos 3 model that needs to be revised to achieve the desired compliance.

- The entire revision phase is composed by a loop that takes into consideration the norms that did not pass the Compliance Evaluation in the previous step. First, for each norm n_i, a revision strategy is chosen depending on whether the violation was caused by the reserve schema, or whether there is enough evidence in the model in order to perform a compliance assessment.

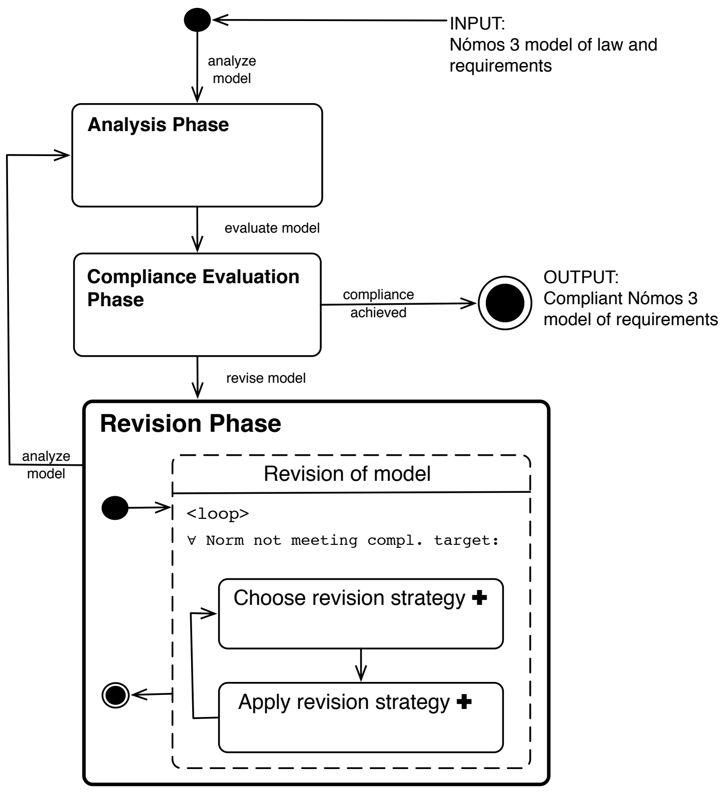

INPUT:
Nómos 3 model of law and requirements
analyze model
Analysis Phase
evaluate model
Compliance Evaluation Phase
compliance achieved
OUTPUT:
Compliant Nómos 3 model of requirements
revise model
Revision Phase
analyze model
Revision of model
<loop>
∀ Norm not meeting compl. target:
Choose revision strategy ✚
Apply revision strategy ✚

Figure 6.4: Workflow detailing the activities in the revision phase of the compliance process.

- Then the revision strategy needs to be applied in the model. In this activity the analyst is guided to perform different activities — like revise or add a goal — in order to solve the compliance problem at hand, with the suggestions offered by the Nòmos 3 reasoning algorithms. If the revision is not successful, another strategy can be chosen and the two activities are repeated until the model has been revised enough to meet the desired compliance value for the norm. When the revision is successful, the next norm the not pass the Compliance Evaluation is considered.

In figure 6.5 we have represented the detailed flow for choosing and applying a revision strategy.

The activity **Choose revision strategy** starts with a series of questions regarding the desired compliance value of the norm n_i considered. (a) If the desired value was *vio* was not met, then the norm is inconclusive. Since a norm is evaluated to inconclusive when its applicability is unknown, more evidence about the domain needs to be gathered/specified in order to definitively evaluate whether the norm applies or not.

- **Gather/specify information.** The objective of this activity is to help the analyst identify situations in the domain that would help clarifying whether a norm applies or not. The Nòmos 3 reasoning algorithm `applicability-search()` will help the analyst identify

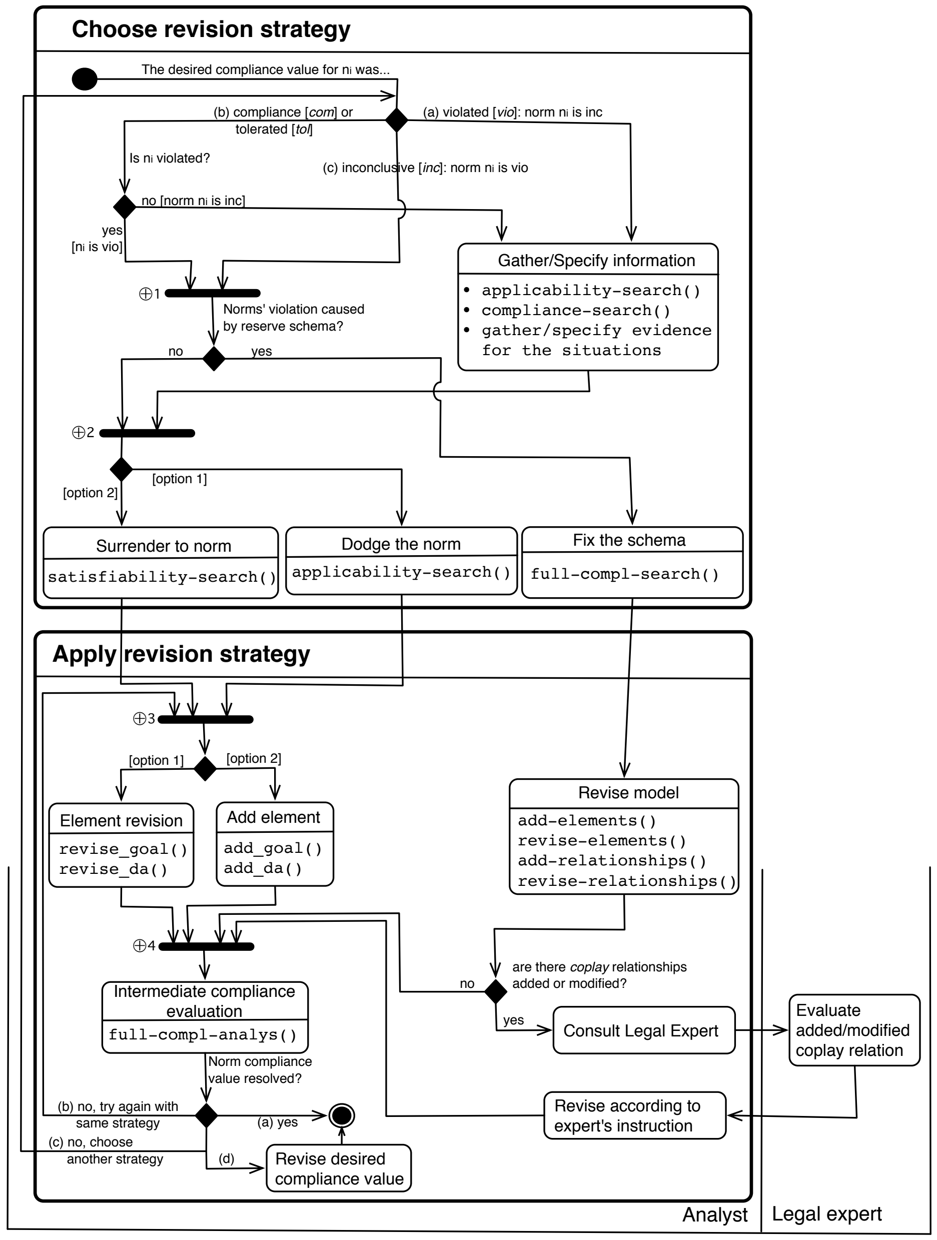

Choose revision strategy
The desired compliance value for ni was...
(b) compliance [com] or tolerated [tol]
(a) violated [vio]: norm ni is inc
Is ni violated?
(c) inconclusive [inc]: norm ni is vio
no [norm ni is inc]
yes [ni is vio]
Gather/Specify information
• applicability-search()
• compliance-search()
• gather/specify evidence for the situations
⊕1
Norms' violation caused by reserve schema?
no
yes
⊕2
[option 1]
[option 2]
Surrender to norm
satisfiability-search()
Dodge the norm
applicability-search()
Fix the schema
full-compl-search()
Apply revision strategy
⊕3
[option 1]
[option 2]
Element revision
revise_goal()
revise_da()
Add element
add_goal()
add_da()
Revise model
add-elements()
revise-elements()
add-relationships()
revise-relationships()
⊕4
are there coplay relationships added or modified?
no
yes
Consult Legal Expert
Evaluate added/modified coplay relation
Intermediate compliance evaluation
full-compl-analys()
Norm compliance value resolved?
(b) no, try again with same strategy
(c) no, choose another strategy
(a) yes
(d)
Revise desired compliance value
Revise according to expert's instruction
Analyst
Legal expert

Figure 6.5: Workflow detailing the activities of choosing and applying a revision strategy in the revision phase.

the situations that make n_i applicable or not. After this analysis different options for compliance are evaluated (join point ⊕2).

(b) If the desired value was *com*/*tol*, then two cases are distinguished: whether the obtained compliance value for the norm was *inc* or *vio*. In the first case, the analyst is lead to the activity "Gather/specify information" which handles cases of inconclusive norms. In the latter case, the process continues to further distinguish which type of violation (join point marked with ⊕1).

Similarly, (c) if the desired value was *inc* and the norm n_i is therefore violated, the process also merges with the case before (join point marked with $\oplus 1$). The join point $\oplus 1$ converges all the cases in which the norm is evaluated as *vio*, and distinguish the cases in which the violation originated from the reserve schema or not. In the first case only one strategy is possible (Fix the schema). The latter case two possible strategies are viable (option 1 and 2), which are also the same strategies available after gathering information for an inconclusive norm (join point $\oplus 2$).

- **Fix the schema.** This first strategy comprises actually all the different possible strategies that can be used to solve a violation of the reserve schema. The algorithm `full-compl-search()` evaluates and classifies the different violation of the schema and suggests the changes to the model that need to be implemented solve the violation (see section 5.2.2).
- **Dodge the norm.** This strategy takes a lateral approach to compliance and uses Nòmos 3 to identify how to avoid complying with the norm ('dodging' the norm). The backward analysis algorithm `applicability-search()` allows the analyst to analyze the legal model and identify which situations make the norm not applicable.[1]
- **Surrender to norm.** This strategy has instead a direct approach to compliance and uses Nòmos 3 to identify how to comply with the norm (therefore 'surrendering'). The backward analysis algorithm `satisfiablity-search()` allows the analyst to analyze the legal model and identify which situations make the norm satisfied (and therefore complied with).[2]

The activity **Apply revision strategy** distinguish the strategies according to the one chosen. If the strategy involved revising the relationships in the model to fix a violation derived from the reserve schema ('Fix the schema'), then the application of this strategy moves directly to its implementation:

- **Revise model.** The objective of this activity is to implement the mitigation strategy that the `full-compl-search()` algorithm indicates for solving the violation. As we saw in section 5.1.2 a violation of the reserve schema can occur for a number reasons, all of which are a symptoms that a role is not entitled to achieve a goal assigned to it. The revision of the model therefore consists in basic operations of adding and removing relationships related to: goals assigned to social roles, delegations of goals between social roles, operationalization of goals, and possibly also relationships linking a social role to the correct legal role. Liking social and legal roles can be a delicate aspects that is not supported by our framework: so if there are modifications done to the existing *coplay* relations, then a

[1]This algorithm explores all the different possibilities for making the norm n_i not applicable: there can be situations that directly affect the norm's applicability, and there can be situations that instead affect a different norm n_j which in turn affect the norm n_i. Because of this possible intertwined network of norms, the revision phase is necessarily followed by an analysis and compliance evaluation to avoid that the revision of some violations leads to the introduction/violation of new norms.

[2]Also in this case, the algorithm explores all the different possibilities for making the norm n_i satisfied, which include satisfying a different norm n_j that in turns makes the norm n_i satisfied.

legal expert should be consulted and the analyst should revise the relationships according to the instruction received. If no corrections are done to *coplay* relations, then the revision of the relationship is considered sufficient. Either way after the revision, the model has to undergo an intermediate compliance evaluation (join point $\oplus 4$).

If the strategy chosen did involve revising relationships for fixing the reserve schema, then both strategies 'Surrender to norm' and 'Dodge the norm' are applied similarly though with different purposes related to their specific analysis (join point $\oplus 3$). Applying the chosen strategy might involve

- **Element revision.** This activity considers the analysis performed in the chosen strategy (studying the satisfiability or applicability of the norm) and revise the existing elements of the model. The reasoning algorithms returns a set of situations that needs to be brought about in the model, and the analyst can for example choose to revise a goal or domain assumption and enrich it with more details satisfying the necessary situations.
- **Adding element.** This activity considers the analysis performed in the chosen strategy (studying the satisfiability or applicability of the norm) and adds an elements of the model. The reasoning algorithms returns a set of situations that needs to be brought about in the model, and the analyst can choose to add a goal or domain assumption satisfying the situations highlighted by the analysis.

After the revision has been implemented, all strategies and options are joined ($\oplus 4$) and an intermediate compliance evaluation is performed. If the desired compliance value for n_i is obtained (a), then the process ends, otherwise the analyst can either: (b) continue revising the model with the same strategy and go back to join point $\oplus 3$, or (c) change strategy and go back at the beginning of the previous activity, or lastly (d) revise the desired compliance value for the norm n_i.

6.3 Conclusions

In this chapter we have presented a systematic processed aimed at helping the analyst evaluate the compliance of a set of requirements with a given piece of law, and amend it to comply with applicable laws. The main objective of this process is to address the knowledge gap between the analyst and the legal issues that he/she needs to deal with during the design of the system-to-be. The proposed Nòmos 3 framework is used in the proposed compliance process to answer important questions about the status of the requirements and the norms (satisfied, applicable, complied with, ...).

One of the main novelty about our process is the flexibility that can be achieved when discussing what compliance is. Sometimes companies may in fact be prone to make compromises between the extent of what law wants the software todo and how much they are willing to change the software to meet the demanded objective. In this light, our process allows the analyst to be flexible and revise the requirements model according to the desired compliance target.

First the analyst is guided through a preliminary analysis of the model, where applicable norms and norms applying to roles are evaluated (section 6.2.1). This type of analysis allows the analyst to perform some small adjustments to the model according this preliminary assessment of the applicable norms.

In the second phase the actual evaluation of compliance is performed, and the desired compliance values are compared against the values obtained in the model (section 6.2.2). When the desired compliance is achieved for all the norms in the model, the process terminates and returns as output the requirements model evaluated, otherwise the model is passed on to the next phase.

In the last revision phase the model is amended according to different strategies that the analyst is guided into choosing (section 6.2.3). The compliance value of a norm in the model and its desired value are the two important information that allow the process to distinguish the different strategies with which the model can be revised. Once the revision strategy is chosen and applied for all norms that did not meet the desired compliance, the process cycles back to the analysis phase before the model is further evaluated for compliance.

Chapter 7

Evaluation of the Nòmos 3 language and reasoning

In this chapter we address the research question **[RQ4:]** *How well does the proposed framework performs when applied to realistic settings?*.

In the first part of the chapter (section 7.1) we show the results of an initial analysis aimed at identifying and analyzing the elements of the law that contribute to its variability. In the second part of the chapter (section 7.2) we apply our Nòmos 3 modeling language and reasoning algorithms to an illustrative case study.

7.1 An analysis of the Italian Guidelines on EHR with Nòmos

In this section we present a preliminary analysis of the "Italian Guidelines on Electronic Health Records and Health Record" that was performed at the very early stages of our research.

The purpose of this analysis was to evaluate the existing concepts in the previous Nòmos modeling language (see section 2.1.1), and identify in the legal document useful concepts that were needed in the new modeling language that was being developed. We choose to analyze this legal text with the Nòmos modeling language because our experience with the language did not require any additional training to perform the classification.

The analysis of this legal text was to be performed manually, so the size/portion of the legal document analyzed was very important. The "Italian Guidelines on the EHR and HF" was found to be an interesting document to analyze given its manageable size (approximately 10 pages) and fairly technical orientation. These guidelines[1] have been issued by the Italian Data Privacy Authority (*Garante*) in 2009 given the lack of specific regulations in the matter of electronic

[1] `http://www.garanteprivacy.it/web/guest/home/docweb/-/docweb-display/docweb/1672821`, accessed July 2015

medical information at the time.[2] The analysis was performed on the official and authenticated English translation of the Guidelines.[3]

Italian laws and legal documents are usually cited and addressed with references to their sections and specific paragraphs. Given the simple structure of this Guidelines — composed only by 10 sections and a total of 91 paragraphs — we refer to a paragraph of a specific section in the form '`§Section.Paragraph`'.

7.1.1 Analysis method

The purpose of our analysis was to evaluate the existing concepts of the Nòmos modeling language and identify other possible useful concepts (see section 2.1.1). Consequently our process was composed of two phases that we manually performed in each paragraph of the law.

1. First we identified in the paragraph the two existing concepts of the Nòmos language: Normative Proposition (NP) and Legal Subject. For elaborated and long paragraphs, we identified multiple NPs and Legal Subjects. Since the objective of this evaluation was not the Hohfeldian classification of rights used in Nòmos,[4] we did not classify further the NP.

2. Once the two existing concept were identified, we classified the remaining part of the paragraph based on how it influenced those two concepts. This last exploratory step was aimed at reflecting how the to the NP or Legal Subject (adding conditions, motivations, examples, ...) were related to the unclassified text. Throughout the analysis we identified the following categories and questions representing them:

   - *Is the unclassified text referencing another law and expanding the domain of the guideline?* We classified these instances as **Extension of domain**.
   - *Is the unclassified text explaining why a certain NP applies?* We classified these instances as **Reason**.
   - *Is the unclassified text explaining the purpose of a NP?* We classified these instances as **Purpose**.[5]
   - *Is the unclassified text offering a practical example?* We classified these instances as **Example**.

[2]In the second half of 2012, a new law had been published to cover the topic, and the guidelines have been adapted consequently (see `http://www.garanteprivacy.it/web/guest/home/docweb/-/docweb-display/docweb/4084632`, accessed July 2015, Italian only).

[3]`http://www.garanteprivacy.it/garante/doc.jsp?ID=1672821`.

[4]`http://plato.stanford.edu/entries/rights/#2.1`, accessed July 2015

[5]We distinguish between Reason and Purpose because of the different terminology used in the legal text. In fact, in some paragraph the legal text was stating exactly the term *purpose* for identifying the allowed-not-allowed operations. In other cases the allowed-not-allowed operations were explained in a more general causal way — e.g., *because of X, then Y* — so we deemed appropriate to keep the two concept separated.

- *Is the unclassified text detailing the property of some document/resource?* We classified these instances as **Properties of resources**.
- *Is the unclassified text explaining when a NP is applicable?* We classified these instances as **Conditional Situation**.

7.1.2 Example of analysis

In the following we report an example of the analysis using section 4, paragraph 1 [§4.1] of the guidelines:

Ex. 1 *Processing of personal data via an EHR/HF [...] should only be performed by health care practitioners*$_1$ *— which does not include technical experts*$_2$*, [...]. Nor are the medical staff*$_3$ *acting in their capacity as forensic medicine experts included — e.g. when examining an individual to establish whether he or she is fit to work and/or drive. The underlying rationale is that the functions discharged by the said professionals [...] are not aimed at treating the data subject, but rather [...]* [§4.1]

In the first phase of the analysis we first identified three Legal Subjects (underlined): health care practitioners, technical expert, and medical staff. Subsequently we identified the one NP (dotted line): "processing of personal data via an EHR/HF".

In the second phase of the analysis we evaluated and classified how the rest of the paragraph influenced this NP. While the technical expert were 'simply' excluded from the rule — they can not process personal data — the medical staff was actually further characterize to specify when they can or can not process. In fact the medical staff — being a health care practitioner — has the right to "Process medical data" *unless* he/she is "acting in their capacity as forensic medicine experts". In this case they do not have the right to process the data. In this example this part (wave underlined) constitute what we call "Conditional situation", i.e. a circumstance that has an impact on the applicability of another circumstance or NP (e.g. the Legal Subject does no longer have a right).

The following second part of the sentence provides an example of what it means, for the medical staff, to act as forensic expert: "when examining an individual to establish whether he or she is fit to work and/or drive". This text (dotted underline) therefore offers a practical example of the rule, and is therefore classified as "Example".

The last sentence of the paragraph ("*The underlying rationale is that ...*") is a piece of text intended to motivate and explain the newly introduced NP, and we therefore classify this part in the category of "Reasons".

7.1.3 Results

The objective of our study was to evaluate how well the Nòmos concepts covered the legal document, and to identify which other relevant concepts would be useful for modeling the law.

Guidelines on EHR/HF		Nòmos [Siena, 2010]		Category	
Paragraphs	**91**	**Normative Proposition**	**111**	**Extension of domain**	**10**
NP-able paragraphs*	72	NP is extension of domain	4%	NP is extension of domain	50%
Not NP-able paragraphs	19	**Legal Subject**	**23**	**Purpose**	**9**
Introduction	53%	*Legal Subject is EHR*	25%	**Reason**	**15**
Examples	31%	*Legal Subject is Data Subject*	22%	**Example**	**34**
Extension of domain	16%	*Legal Subject is Data Controller*	14%	*with compliance meaning*	35%
		Legal Subject is other (20 Subjects)	39%	*with no compliance meaning*	65%
		Holder is not explicit	13%	**Properties of resources**	**6**
		Correlative not expressed	92%	**Conditional Situation**	**41**

* Paragraphs that carried at least one NP.

Table 7.1: Numerical results of the analysis of the Italian Guidelines on Electronic Health Records with Nòmos [Siena, 2010]

Our analysis consisted in manually classify all the parts of the legal document according to the method describe above. In order to evaluate both the coverage of existing concept and identify new ones, we have analyzed the number of occurrences in each category.

Table 7.1 summarizes the numerical results of our study. In the left column of the table we report some statistical data about the Guidelines. In the middle column we show the numerical data describing the identified Nòmos concepts. In the right column we report the data regarding the six categories of our classification.

Left column: data about guidelines

The legal document analyzed consisted of 91 paragraphs that were analyzed using the method described above. Not all paragraphs of the law had normative content, i.e. provided some rights/duties for some subjects. We therefore labelled a paragraph as 'NP-able' or 'not NP-able' depending on whether it was possible to identify a NP or not.

As we can see from table 7.1, out of the 91 paragraphs, approximately 80% were NP-able. The remaining not NP-able paragraphs — paragraph not originating any NPs — were mainly introductory (53%), or paragraphs with dedicated examples (31%), or with a long dedicated list of extensions of domain (16%).

Middle column: results about Nòmos

Normative Proposition (NP). As you can see from table 7.1 we have identified a total of 111 NPs. A little less than 5% of these NPs were cases in which the NP was an obligation to comply with another law. For example:

Ex. 2 *[. . .] a data controller should comply with the legislation protecting anonymity of individuals [. . .]* [§5.6]

These instances are worded as a duty for a specific Legal Subject (the data controller), but were actually extending the scope of the current legal document to another law. Consequently we classified these cases as NPs for that Legal Subject and also as instances of the category of "Extension of domain".

Another interesting trait we noticed is that the law does not simply refer to rights and duties but also to their exercisability. For example:

Ex. 3 *[. . .] the data controller is free to require that data subjects exercise the said [blanking] right in the presence of the physician [. . .]* [§3.14]

The NP contained in this sentence is the right to "Require the Data Subject to exercise his right in presence of a physician", so in the Hohfeldian taxonomy it is actually the power to constraint the exercisability of somebody else's right.

In addition to the previous two cases, another important fact we noticed is that several NPs kept on referring to the same action/activity. For example, the NP "Process personal data" is spread among many NPs for several Legal Subjects: Health Care Practitioners can "process personal data" (example 1), Technical Experts must "not process personal data" (example 1), the Data Controller has to "*appoint the natural person in charge of this processing* [of personal data]" [§4.4], and so on. This "processing of personal data" is the common denominator of these and other NPs, and the redundancy around it is what creates the variability for this core action. By capturing this action separately into multiple NPs, we do not capture this common thread, making it more difficult to identify the variability related to this event, i.e., how performing this action in different contexts makes different NP applicable or not.

Legal Subjects. In the guidelines we have identified a total of 23 Legal Subjects. Not surprisingly the Legal Subjects that were regulated the most (61%) were the "EHR system", the "Data Subject" and "Data Controller" (see table 7.1). The remaining 40% of the NPs were addressed to other 20 Legal Subjects.

Three main peculiarities arose while identifying the subjects addressed by the NP. First, in the 12% of the identified NPs, the (direct) Legal Subject was not specified in the text. For example:

Ex. 4 *[. . .] The appropriate measures should be taken to allow tracing back the entities responsible for creating and collecting the data [. . .]* [§2.6]

In this example the law does not explicitly say who should take the appropriate measures. In the particular case of these guidelines, we have identified that all these cases of unspecified Legal Subjects referred to the EHR system. However it could be the case that, from a legal point of view, if someone else fulfills these duties compliance might still be drawn.

Secondly — on a related issue — we have found a substantial number of instances (91%) where the correlative part of the right/duty was also not specified. For example:

Ex. 5 *The consent must be given on a separate, specific basis [. . .]* [§3.7]

The legal context is obvious enough to interpret that the Data Subject is the Legal Subject who should express the consent, however it is not specified to whom they have to give it to, or who gives the Data Subject the consent-form (and nowhere in the text these specifications were given).

Lastly, it is worth noticing how the Legal Subjects in the law can refer both to general and abstract classes (e.g. "*Entities co-ordinating EHR projects that covers a small area*" [§9.4]) as well as to specific roles ("*the health care practitioner that is or will be treating the Data Subject*" [§5.12]). It is clear how the two Legal Subjects belong to different conceptual levels: a general group of legal entities on one side, and the specific practitioner treating a specific Data Subject on the other.

Right column: results about our classification

Extension of domain. This category identifies a piece of text that references another law applying to the context and that needs to be complied with. The instances of this category were generally trivial to identify, as in the case of example 2. As we mentioned before, in this example the extension of domain is specifically targeted to a Legal Subject. These cases were the extension of domain was formulated as a duty for a specific Legal Subject, corresponded to 50% of all the identified instances in this category. The other 'simpler' instances were cases like "*The legislation on access to administrative records (. . .) is obviously left unprejudiced*" [§5.19]. In general, approximately 10% of the NPs contained an extension of domain.

Reason. This category represents the part of the legal text identifying motivations for a NP to be in place. Instances of this category answer the question 'why does a certain NP apply?'. Sometimes in fact the legal text provides in writing the reasoning behind the existence of NP, and this category identifies these cases. For example:

Ex. 6 *Since the medical data and documents contained in a EHR are collected from different sources* [the reason]*, take appropriate measures* [the NP] *to allow tracing back the entities responsible for creating the data [. . .]* [§2.6]

In this paragraph, the fact that the medical data are collected from multiple sources is not doubted by the lawmaker, who would have otherwise started the sentence using an "if"-condition (therefore considering the possibility of not collecting data from multiple sources). What they are giving is therefore an explanation about why the following NP is in place. In general, instances of this category appear in approximately 14% of the NPs we have identified.

Purpose. This category identifies the part of the legal text where it is stated explicitly the purpose or the aim of a specific NP. This guidelines deal with issues about privacy of personal and sensitive information, so it is not surprising to find paragraphs where some actions are allowed only for some purposes. For example:

Ex. 7 *The medical data contained in an EHR/HF may only be transferred abroad for purposes of prevention, diagnosis and treatment [. . .]* [§7.2]

If administrative purposes are to be also achieved via EHRs and/or the HF [. . .] keep administrative data separate from medical information. [§2.10]

In this extract it is clear how the purpose for which an action is taken — prevention, diagnosis, administrative — defines and limits the scope of the rule — when data can be taken abroad. In general, instances of this category appear in approximately 8% of the NPs we have identified.

Example. An interesting trait arose from these guidelines where we have found a significant amount of examples where the legal text offered answers/suggestions regarding compliance. We classified as "examples" all the instances where the legal text was providing further clarifications on a given activity or NP. Furthermore we classified all these instances in two sets discerning whether the example had a strong compliance meaning — i.e. the law was providing details on who to comply — or not. For example:

Ex. 8 *A few EHR projects taken into consideration empower Data Subjects to "blank" information via a sealed electronic envelope, which is [. . .]; alternatively, random codes are allocated to individual events, which prevents establishing links between certain items of flagged information.* [§3.12]

This paragraph represents an Example with strong compliance meaning. The law basically suggests to use a solution — the "sealed electronic envelope" — to comply with the 'blanking right'. Also, the sentence continues clarifying the solution and providing further explanations why it respects the right of the Data Subject. We found examples with strong compliance meaning in 35% of all examples.

Other instances of examples (the remaining 65%) were clarifications on information, such as "*[. . .] past clinical events (e.g. previous medical reports)*" [§3.15]. This type of examples also plays an important role as it provides a "certified" interpretation of the terms and situations addressed by the law.

In general, instances of the 'Example' category occurred in approximately 30% of the NPs we have identified.

Properties of resources This category identifies the part of the law where the characteristics of some resources are given and specified as mandatory feature of the resource. For example:

Ex. 9 *The information notice should clearly specify [. . .] the entities (or categories of entities) that [. . .] may access the EHR/HF [. . .]* [§8.4]

The information notice is not a Legal Subject but it is a resources/document that needs to have certain properties: must clearly specify the entities, must be in written form, and so on. Instances of this category appear in approximately 5% of the NPs we have identified.

Conditional Situation This category identifies the part of the law describing a condition on the applicability of a NP.[6] For example, the guidelines state that the health record of a Data Subject can be accessed without his/her consent "*if this is found to be indispensable to protect a third party's and/or the public health*" [§8.6]. When this condition holds, then it triggers a special rule allowing an exception to the disclosure of data without consent. As we can see from table 7.1, the instances of this category are very common in this legal text and occur in approximately 37% of the NPs we have identified.

These conditional situations influence the applicability of NPs under different aspects. They can describe the capabilities of a Legal Subject (e.g. example 1 "when acting as forensic expert"), using these capabilities as condition for the applicability. They can describe the environment (e.g. example 3 talks about the presence of a physician), using that environmental constraint to confine the exercisability of a right. Furthermore, these conditional situations can describe the temporal dimension of the NP (e.g. when saying that "*when the Data Subject becomes of (legal) age*" [§3.18]), and this is what binds the applicability of a certain NP. The instances of this category can indeed model and shape how NPs come into play. They embody the variability related to the application of a NP, of an exception, or of something causing changes in the application of a NP.

[6]Purposes and reasons can also be 'Conditional Situations': in these cases the text is classified as both.

7.1.4 Discussion

During the analysis phase we noticed a patter of problematic traits related to either the analyzed domain or the Nòmos modeling language. In fact, while some categories (e.g. 'Property of resources' and 'Example') are probably interesting peculiarities of these guidelines — or in general of technical regulations — other categories highlighted more general and fundamental issues relevant for the domain of regulatory compliance for requirements engineering.

The concept of NP — as well as the Nòmos relationships modeling priorities among NPs (see section 2.1.1) — tolerates and allows the modeler to represent an unstructured redundancy that does not capture the variability around it. For example in our analysis we have noticed how many NPs gravitate around the "Processing of personal data". Structuring the redundancy around this main part makes it possible to manage the variability related to whether a certain NP applies or not. A simpler element is therefor needed to represent both the core of the NP and the conditions influencing its applicability. An important challenge is to eliminate the uncertainty related to the legal variability, and allow requirements engineers to reason easily about how/when different part of the law apply to a given set of requirements.

During the analysis we have encountered different types of abstractions of the Legal Subjects. Sometimes the legislator regulates specific subjects — very specific roles — like "the Practitioner who is treating the Data Subject". Other times he refers to just "a Practitioner", and some other times the law generally talks about "Medical Staff" or "Employee". In the mind of the legislator there is a hidden implicit role classification (like a role hierarchy), and depending on the intent of the regulation he addresses a more general or specific type of role. The challenge is that this hidden role classification can differentiate what obligations apply or why they don't. The fact that this "implicit hierarchy" can only be perceived and is not explicitly addressed in the regulation, leaves the classification job to those who need this clarification. Mapping this roles classification from the legal domain to the one of your system becomes a delicate issue once again influenced by interpretations. We therefore would envisage the requirements engineer to consult a legal expert when evaluating this mapping.

An interesting pawn in the chessboard of legal variability comes from the category of 'Purpose'. Example 7 introduces the purpose as criterion for the applicability of a duty. Often this purpose shapes the behavior of a Legal Subject, therefore it becomes crucial to design a system allowing an actor to pursue its goals while at the same time limiting the achievement of purposes other than the ones required by law. An interesting challenge is in the identification and evaluation of the purposes of a software system and of its actor, and validate them against the legal ones.

Threats to validity. An important limitation of this analysis comes from the legal document chosen for the study: indeed the Italian Guidelines on EHR regulate a very specific sector. Other laws and more comprehensive regulations — such as the Italian Data Privacy Law, or the European Data Protection Directive — may in fact have different categories than the ones we identified. However, as a preliminary evaluation — especially since the analysis was done manually — we believe the reliability of the results is therefore adequate to our purpose.

Another limitation is given by the fact that this regulation is a relatively short document (91 paragraphs).[7] The numerical results discussed are presented as an indication of the number of instances we found for each category, but are not aimed at characterizing a general distribution. As a consequence, w.r.t. a generalizability issue, we consider the results of this investigation as first preliminary assessment of the Nòmos framework that needs to be confirmed by further analysis on different documents. Moreover — being our analysis unrelated to a pure ontological characterization but rather to a preliminary investigation of elements influencing the effect of NPs — we believe that the categories are adequate to the limited scope of our analysis. A more thorough evaluation of the categories and their influence on NPs will be part of our future work.

7.2 Evaluation with illustrative example

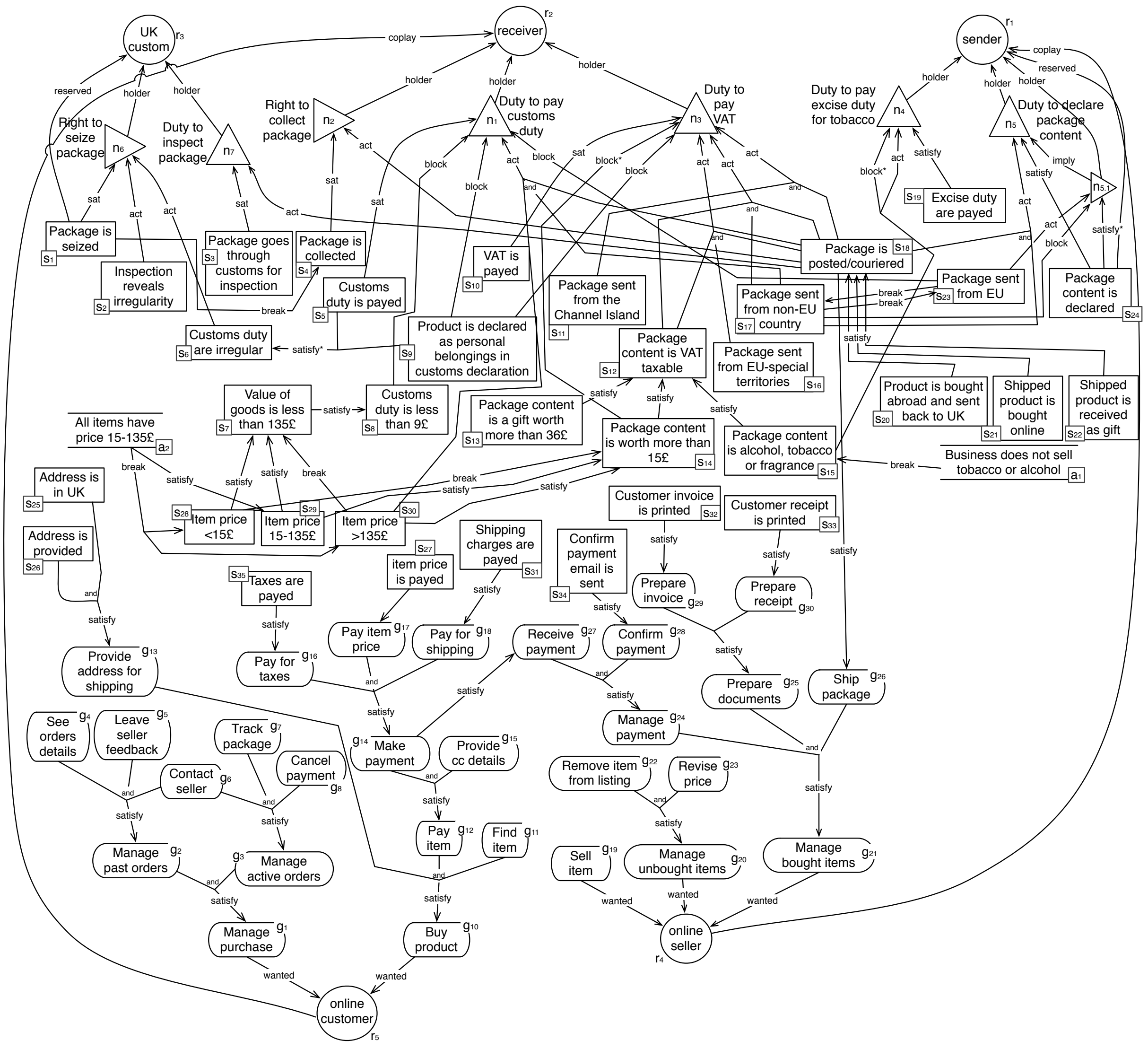
In this section we evaluate the reasoning algorithms presented in chapters 4 and 5 using an illustrative example, addressing the last research question **[RQ4:]** *How well does the proposed framework performs when applied to realistic settings?*.

In this section we maintain the running legal context of the British legislation regulating tax and customs of goods sent from abroad.[8] The example we use is that of a software for selling goods online (like eBay) where the seller sells products in U.K. and ships them from warehouses located in different parts of the world. The goal of the software is to help the seller manage his goods and automatically prepare the necessary paperwork for correctly shipping his goods to U.K. and identify the right taxes that need to be payed from the buyer or in addition to the item price. Figure 7.1 shows the Nòmos 3 model with both the legal and requirements model for the online seller example.

Legal context. The legal context analyzed for this software regards the U.K. legislation regulating tax and customs of goods sent from abroad.[9] The law mandates that "*Anything posted or couriered to you goes through customs to check it isn't banned or restricted and you pay the right tax and 'duty' (customs charges) on it*", so it is important that the software managing shipping of goods correctly alerts the seller and buyer of taxes to be payed. Moreover, if the shipped package contains alcohol or tobacco, the law mandates the sender to pay excise duty, and when goods is shipped from outside the EU, the seller also has to correctly declare goods to avoid the package being seized.

The legal model representing the extract of the regulation is presented in the top part of figure 7.1. Starting from the left side of the figure, the U.K. custom has the right (n_6) to seize

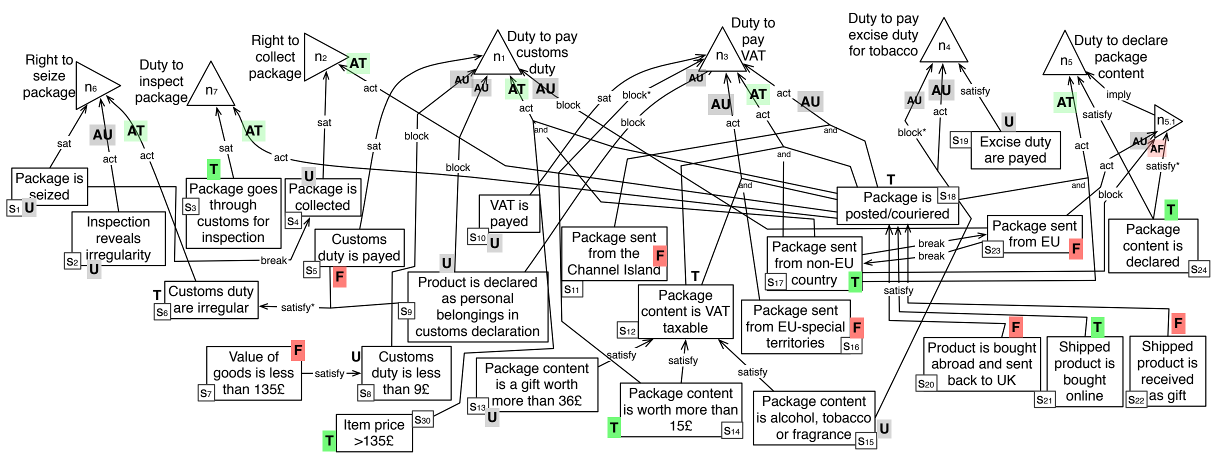
[7] For comparison, the Italian Data Privacy Law has 540 paragraphs, without the appendices.
[8] https://www.gov.uk/goods-sent-from-abroad/overview.
[9] https://www.gov.uk/goods-sent-from-abroad/overview.



UK custom r3
receiver r2
sender r1
coplay
holder
reserved
Right to seize package n6
Duty to inspect package n7
Right to collect package n2
Duty to pay customs duty n1
Duty to pay VAT n3
Duty to pay excise duty for tobacco n4
Duty to declare package content n5
n5.1
sat
act
block
block*
satisfy
satisfy*
imply
break
and
Package is seized s1
Inspection reveals irregularity s2
Package goes through customs for inspection s3
Package is collected s4
Customs duty is payed s5
Customs duty are irregular s6
Value of goods is less than 135£ s7
Customs duty is less than 9£ s8
Product is declared as personal belongings in customs declaration s9
VAT is payed s10
Package sent from the Channel Island s11
Package content is VAT taxable s12
Package content is a gift worth more than 36£ s13
Package content is worth more than 15£ s14
Package content is alcohol, tobacco or fragrance s15
Package sent from EU-special territories s16
Package sent from non-EU country s17
Package is posted/couriered s18
Excise duty are payed s19
Product is bought abroad and sent back to UK s20
Shipped product is bought online s21
Shipped product is received as gift s22
Package sent from EU s23
Package content is declared s24
Address is in UK s25
Address is provided s26
item price is payed s27
Item price <15£ s28
Item price 15-135£ s29
Item price >135£ s30
Shipping charges are payed s31
Customer invoice is printed s32
Customer receipt is printed s33
Confirm payment email is sent s34
Taxes are payed s35
All items have price 15-135£ a2
Business does not sell tobacco or alcohol a1
Manage purchase g1
Manage past orders g2
Manage active orders g3
See orders details g4
Leave seller feedback g5
Contact seller g6
Track package g7
Cancel payment g8
Buy product g10
Find item g11
Pay item g12
Provide address for shipping g13
Make payment g14
Provide cc details g15
Pay for taxes g16
Pay item price g17
Pay for shipping g18
Sell item g19
Manage unbought items g20
Manage bought items g21
Remove item from listing g22
Revise price g23
Manage payment g24
Prepare documents g25
Ship package g26
Receive payment g27
Confirm payment g28
Prepare invoice g29
Prepare receipt g30
wanted
online customer r5
online seller r4

Figure 7.1: Example of a Nòmos 3 model — including both legal and requirements model — for tax and customs for goods shipped to U.K. in the example of a software for online sellers.

the package when an irregularity is revealed during an inspections or when the custom duty is not payed for a package that is not personal belongings ($s_5 \wedge s_9 \xrightarrow{\text{satisfy}*} s_6$). The U.K. custom has also the duty to inspect packages shipped to U.K. (n_7). In the analyzed legal context the receiver of the package has the duty to pay VAT tax (n_3), to pay custom duty (n_1), and the right to collect the package (n_2). Both duties are satisfied when the respective tax is payed and they apply depending on whether the package is shipped from a non-EU country or whether the product is declared as personal belongings ($s_9 \xrightarrow{\text{block}} n_1$). Similarly, when the duty to pay is less than £9, the receiver does not have to pay ($s_8 \xrightarrow{\text{block}} n_1$). VAT must be payed on *goods sent from non-EU countries and EU special territories (e.g. the Canary Islands) if* the content is VAT



Right to seize package
n6
Duty to inspect package
n7
Right to collect package
n2
Duty to pay customs duty
n1
Duty to pay VAT
n3
Duty to pay excise duty for tobacco
n4
Duty to declare package content
n5
n5.1
Package is seized
Inspection reveals irregularity
Package goes through customs for inspection
Package is collected
Customs duty is payed
Customs duty are irregular
VAT is payed
Product is declared as personal belongings in customs declaration
Package sent from the Channel Island
Package content is VAT taxable
Package sent from non-EU country
Package sent from EU-special territories
Package is posted/couriered
Package sent from EU
Excise duty are payed
Package content is declared
Value of goods is less than 135£
Customs duty is less than 9£
Item price >135£
Package content is a gift worth more than 36£
Package content is worth more than 15£
Package content is alcohol, tobacco or fragrance
Product is bought abroad and sent back to UK
Shipped product is bought online
Shipped product is received as gift
sat
act
block
block*
satisfy
satisfy*
break
imply
and
AU
AT
AF
U
T
F

Figure 7.2: The Nòmos 3 legal model for the running example of tax and customs for goods sent to U.K. from abroad (https://www.gov.uk/goods-sent-from-abroad/overview): `applicability-analysis()`.

taxable ($(s_{18} \wedge s_{17} \wedge s_{12}) \xrightarrow{\text{activate}} n_3$), so when *it contains: (a) gifts worth more than £36, (b) other goods worth more than £15, or (c) alcohol, tobacco products and fragrances of any value* ($s_{13} \xrightarrow{\text{satisfy}} s_{12}, s_{14} \xrightarrow{\text{satisfy}} s_{12}, s_{14} \xrightarrow{\text{satisfy}} s_{12}$). All goods sent by mail order from the Channel Islands is subject to VAT taxes regardless from their value ($(s_{18} \wedge s_{11}) \xrightarrow{\text{activate}} n_3$). In case the product is declared as personal belongings (s_9), VAT does not need to be payed ($s_9 \xrightarrow{\text{block}} n_3$), and the duty to pay VAT is also not applicable if the package content is worth less than £15 ($s_{14} \xrightarrow{\text{block}^*} n_1$). The sender has the duty to pay "*Excise Duty on any alcohol or tobacco sent*" ($s_{15} \xrightarrow{\text{activate}} n_4, s_{19} \xrightarrow{\text{satisfy}} n_4$), and excise duty does not need to be payed when the package content is not alcohol or tobacco ($s_{15} \xrightarrow{\text{block}^*} n_4$). Moreover the sender "*must declare goods correctly if they're sent from outside the EU*" ($(s_{18} \wedge s_{17}) \xrightarrow{\text{activate}} n_5$), but if they are not sent from outside the EU there is no need to declare the goods ($s_{17} \xrightarrow{\text{activate}^*} n_{5.1}$ and $n_{5.1} \xrightarrow{\text{imply}} n_5$).

7.2.1 Reasoning with legal variability

In chapter 4 we introduced the algorithms that explore the variability of Nòmos 3 legal models. Reasoning in Nòmos 3 is performed by means of backward and forward reasoning algorithms that exploit the satisfaction value of Situations to evaluate the value of other elements of the model. Forward reasoning involves assigning a satisfaction value to the elements of the model and propagate them across the model according to the semantics of the relations. Examples of forward reasoning are applicability analysis (section 4.1.2), satisfiability analysis (section 4.1.4), and compliance analysis (section 4.1.5). Backward reasoning involves instead assigning a desired value to some target elements and explore the model for possible satisfaction value of Situations. Example of backward reasoning are applicability search (section 4.1.1), satisfiability search (section 4.1.3), and compliance search (section 4.1.6). In the following we analyze the algorithm evaluating the applicability of a set of Norms in a given scenario (applicability analysis), and the algorithm that searches for an assignment to the Situations in the model in order to obtain the desired compliance value for some Norms (compliance search).

Applicability analysis. In figure 7.2 we represented the legal model for the online seller example, and in the following we will consider in input to the algorithm `applicability-analysis()` this Nòmos 3 legal model, and the satisfaction value for the Situations describing the following scenario. A British online seller has sold online an item for £160 ($s_{30}, s_{14} = \text{ST}, s_7 = \text{SF}$) and ships to U.K. ($s_{21} = \text{ST}, s_{20}, s_{22} = \text{SF}$) a package from the warehouse in China ($s_{17} = \text{ST}, s_{11}, s_{16} = \text{SF}$). The package content has been declared ($s_{24} = \text{ST}$) and is inspected at customs ($s_3 = \text{ST}$), and custom duty are not payed ($s_5 = \text{SF}$). All the other Situations in the model ($s_{19}, s_{15}, s_{10}, s_{13}, s_9, s_4, s_2, s_1$) are considered as having unknown satisfaction value. The satisfaction value of the Situations is highlighted in figure 7.2. In this example we use the algorithm `applicability-analysis()` to study which Norms are applicable in this current scenario.

- The right n_6 receives two value for applicability from the two incoming relationships: $s_2 \xrightarrow{\text{activate}} n_6$ propagates unknown applicability since $s_2 = \text{SU}$, and $s_6 \xrightarrow{\text{activate}} n_6$ propagates positive applicability since $s_6 = \text{ST}$.[10] The two values are treated as being in disjunction (see table 3.1) and the right n_6 to seize the package is applicable.

- The duty n_7 to inspect the package has only one incoming relationship for applicability: $s_{18} \xrightarrow{\text{activate}} n_6$. The Situation s_{18} has positive satisfiability, as we know from our scenario that a shipped product is bough online ($s_{21} = \text{ST}, s_{21} \xrightarrow{\text{satisfy}} s_{18}$). The duty n_7 therefore receives positive applicability from its incoming relationship and is therefore applicable.

- The duty n_1 to pay custom duty has four incoming relationships for applicability. From the left, the duty receives unknown applicability since it is not known if the tax amounts to less than £9 ($s_8 \xrightarrow{\text{block}} n_1$). Similarly, since it is not known if the product has been declared as personal belongings, n_1 receives unknown applicability ($s_9 \xrightarrow{\text{block}} n_1$). The incoming relationship $s_{30} \wedge s_{17} \wedge s_{18} \xrightarrow{\text{activate}} n_1$ propagates positive applicability since all three Situations are satisfied. Lastly, the relationship $s_8 \xrightarrow{\text{block}} n_1$ propagates unknown applicability since the package is not sent from the EU ($s_{23} = \text{SF}$). The disjunction of the four applicability values ($\text{SU} \vee \text{SU} \vee \text{ST} \vee \text{SU} = \text{ST}$) follows the rule defined in chapter 3 where it is sufficient that at least one of the value is positive (see table table 3.1). As a result, the duty n_1 is applicable.

- The applicability of the right n_2 depends on whether the package has been shipped. Since the Situation s_{18} is satisfied, the relationship propagates positive applicability to the target Norm.

[10] The Situation $s_6 = \text{ST}$ since $s_5 = \text{SF}, s_5 \xrightarrow{\text{satisfy}^*} s_6$.

- The duty n_3 to pay VAT has five incoming relationships characterizing its applicability.

  The first relationship from the left side $s_{14} \xrightarrow{\text{block}^*} n_3$ would propagate false applicability if it was known that the package content was worth less than £15. However since it is known that the content is worth more than £15 ($s_{14} = \text{ST}$) the duty receives unknown applicability from this relationship. The second relationship $s_9 \xrightarrow{\text{block}} n_3$ represents that VAT should not be payed when the product is declared as personal belongings. Since it is not known whether the declaration listed the content as personal belongings ($s_9 = \text{SU}$), the duty receives unknown applicability from this relationship. The third incoming relationship ($s_{12} \wedge s_{16} \wedge s_{18} \xrightarrow{\text{activate}} n_3$) represents that a package shipped from EU special territories with VAT taxable content should pay VAT. This relationship also propagates unknown applicability: the conjunction of the three Situations ($s_{12} = \text{ST}, s_{16} = \text{SF}, s_{18} = \text{ST}$) is SF and the *activate* relationship therefore propagates unknown satisfiability to the antecedent of the Norm, i.e. unknown applicability.

  The fourth relationship ($s_{12} \wedge s_{17} \wedge s_{18} \xrightarrow{\text{activate}} n_3$) propagates positive applicability since all three source Situations are satisfied: a package is shipped from a non-EU country with VAT taxable content should pay VAT. The last relationship $s_{11} \wedge s_{18} \xrightarrow{\text{activate}} n_3$ propagates unknown applicability since the conjunction of the two source Situations is SF ($s_{11} = \text{SF} \wedge s_{18} = \text{ST}$). All five values for applicability values are treated as being in disjunction ($\text{AU} \vee \text{AU} \vee \text{AU} \vee \text{AT} \vee \text{AU}$) and since at least one value is positive, the duty n_3 is evaluated as being applicable.

- The duty n_4 to pay excise duty is targeted by two relationships for applicability. The first from the left represents that excise duty should not be payed when the package content is not tobacco/alcohol/fragrance ($s_{15} \xrightarrow{\text{block}^*} n_4$). The other incoming relationship represents that excise duty should be payed when the content is tobacco/alcohol/fragrance ($s_{15} \xrightarrow{\text{activate}} n_4$).

  Since it is unknown whether the package contains tobacco/alcohol/fragrance ($s_{15} = \text{SU}$), the duty receives unknown applicability from both relationships. Similarly the exception $n_{4.1}$ not to pay for duty receives unknown applicability.

- Lastly, the duty n_5 also has only one incoming relationship for applicability: $s_{18} \wedge s_{17} \xrightarrow{\text{activate}} n_5$ and since both source Situations are satisfied, the duty is applicable. The right $n_{5.1}$ not to declare the package has instead unknown applicability since the package is sent from a non-EU country ($s_{17} \xrightarrow{\text{activate}^*} n_{5.1}$).

Compliance search. In the following we will consider in input to the algorithm `compliance-search()` the same Nòmos 3 model in figure 7.3 and as target value for the Norms we want "compliance" or "tolerated". The algorithm works by exploring possible assignments to

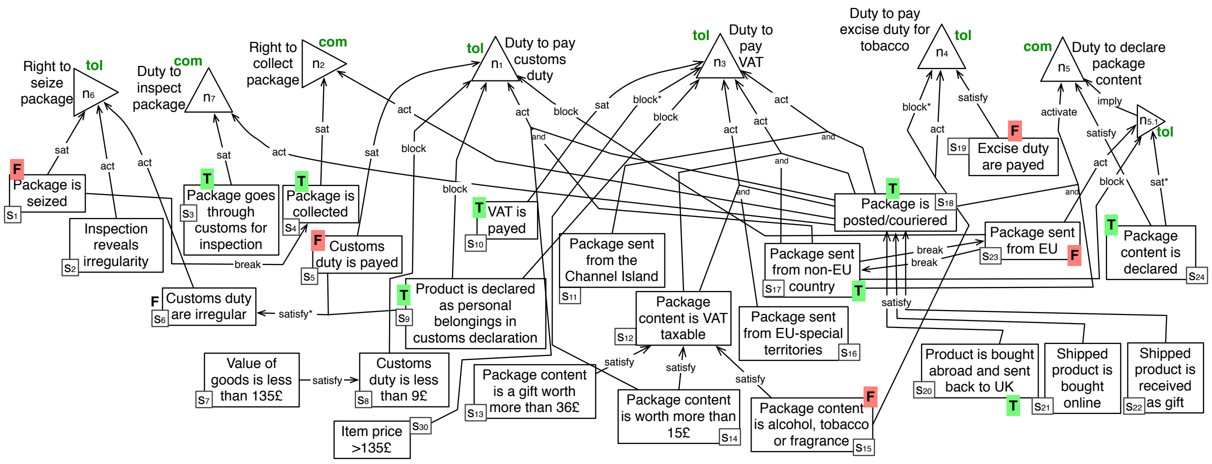

Right to seize package
n6
tol
Duty to inspect package
n7
com
Right to collect package
n2
com
tol
Duty to pay customs duty
n1
tol
Duty to pay VAT
n3
Duty to pay excise duty for tobacco
n4
tol
com
Duty to declare package content
n5
n5.1
tol
F
Package is seized
s1
Inspection reveals irregularity
s2
T
Package goes through customs for inspection
s3
T
Package is collected
s4
F
Customs duty is payed
s5
F
Customs duty are irregular
s6
Value of goods is less than 135£
s7
Customs duty is less than 9£
s8
T
Product is declared as personal belongings in customs declaration
s9
T
VAT is payed
s10
Package sent from the Channel Island
s11
Package content is VAT taxable
s12
Package content is a gift worth more than 36£
s13
Package content is worth more than 15£
s14
F
Package content is alcohol, tobacco or fragrance
s15
Package sent from EU-special territories
s16
Package sent from non-EU country
s17
T
T
Package is posted/couriered
s18
F
Excise duty are payed
s19
Product is bought abroad and sent back to UK
s20
T
Shipped product is bought online
s21
Shipped product is received as gift
s22
Package sent from EU
s23
F
T
Package content is declared
s24
Item price >135£
s30
sat
act
break
satisfy*
satisfy
block
block*
and
imply
activate
sat*

Figure 7.3: The Nòmos 3 legal model for the running example of tax and customs for goods sent to U.K. from abroad (https://www.gov.uk/goods-sent-from-abroad/overview): `compliance-search()`.

the Situations and evaluating whether the values propagated to the target Norm correspond to the desired ones.

- Starting from the right side of the figure, the compliance of the duty n_5 can be achieved by complying directly with the duty, or by complying with the Norm $n_{5.1}$ ($n_{5.1} \xrightarrow{\text{imply}} n_5$). An assignment that would make the Norm complied with is for example one where the Norm n_5 is applicable ($s_{18}, s_{17} = \text{ST}$ and $s_{18} \wedge s_{17} \xrightarrow{\text{activate}} n_5$) and satisfied ($s_{24} = \text{ST}$). With this satisfaction value, the Norm $n_{5.1}$ is evaluated as tolerated. Since there are three possibilities for s_{18} to be satisfied ($s_{20} \xrightarrow{\text{satisfy}} s_{18}, s_{21} \xrightarrow{\text{satisfy}} s_{18}, s_{22} \xrightarrow{\text{satisfy}} s_{18}$) it is sufficient that one of the three Situations is satisfied. The satisfaction values of the Situations are represented in figure 7.3.

- The compliance of the duty n_4 (pay excise duty for tobacco) can be achieved by paying the excise duty when sending tobacco, or by making sure the content is not tobacco/alcohol/fragrance. An assignment that makes the duty n_4 complied with is for example one where the package content is not tobacco ($s_{15} = \text{SF}$ and $s_{15} \xrightarrow{\text{block}} n_4$) and excise duties are not payed ($s_{19} = \text{SF}$ and $s_{19} \xrightarrow{\text{satisfy}} n_4$). The duty is not applicable and not satisfied: it is evaluated as tolerated.

- The compliance of the duty n_3 depends on the Situations ruling its applicability and satisfiability. The current assignment already has $s_{17}, s_{18} = \text{ST}$, so one possibility to make the duty applicable is to consider a Situation where the package content is taxable ($s_{12} = \text{ST}$). In this scenario the duty n_3 is applicable ($s_{17} \wedge s_{18} \wedge s_{12} \xrightarrow{\text{activate}} n_3$) and it is complied with when it is also satisfied, i.e., the VAT tax is payed ($s_{10} = \text{ST}$). Since there are two possibilities for a package to be VAT taxable ($s_{13} \xrightarrow{\text{satisfy}} s_{12}, s_{14} \xrightarrow{\text{satisfy}} s_{12}$),[11] it is sufficient that one of the two Situations is satisfied.

- Achieving compliance of n_1 also depends on its applicability and satisfiability. For example if the item price was higher than £135, the duty would be applicable (s_{30} = ST, and $s_{30} \wedge s_{17} \wedge s_{18} \xrightarrow{\text{activate}} n_1$) and the only way to comply would be to pay custom duty. Another possibility for complying (the one shown in figure 7.3) is to declare the package as personal belongings (s_9 = ST) which makes the duty not applicable and makes not necessary to pay custom duty (s_5 = SF).

- The right n_2 to collect the package is applicable since the package has been shipped. The only possible way to comply with the right (to exercise the right) is to collect the package s_4 = ST. If the package is seized it is however not possible to collect the package ($s_1 \xrightarrow{\text{break}} s_4$), so the solution scenario must include that s_1 = SF.

- The duty n_7 to inspect the package is applicable since s_{18} = ST and $s_{18} \xrightarrow{\text{activates}} n_7$. The only possible way to obtain the desired compliance value is to make the duty satisfied and inspect the package (s_3 = ST, $s_3 \xrightarrow{\text{satisfy}} n_7$).

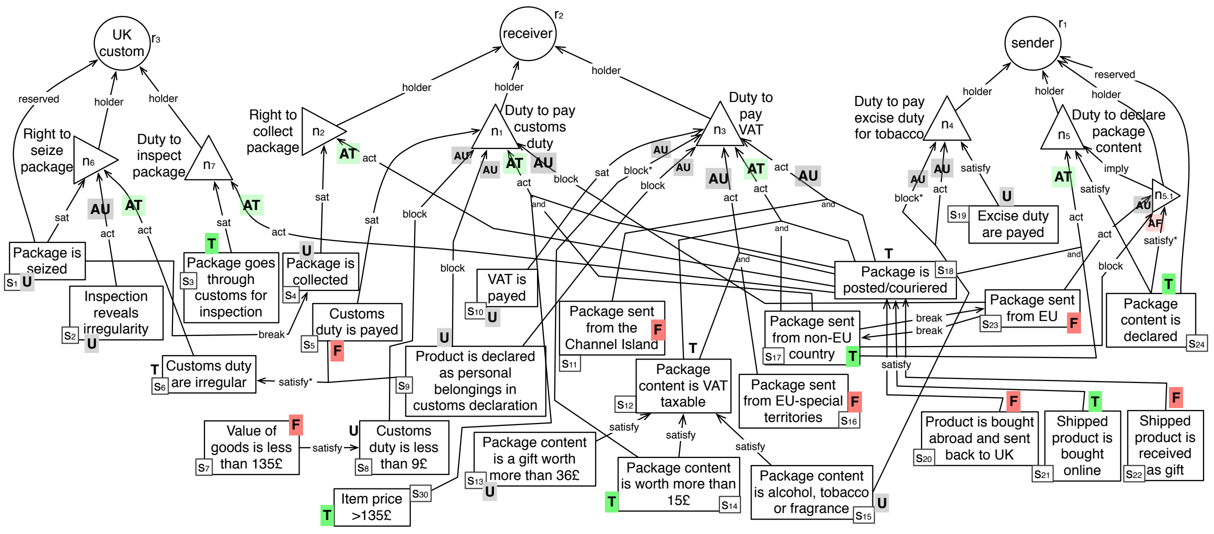
- Lastly, the right to seize the package is only applicable when s_6 holds and custom duty are irregular — and in the current scenario we already know that s_6 = SF — or when the inspection reveals irregularity s_2 = ST. However in the current scenario we already know that the package is not seized (s_1 = SF), otherwise the Situation s_4 representing that the 'package is collected' would not hold. In the current model the right to seize the package can only be applicable when the inspection reveals irregularity (s_2 = ST), but it can not be satisfied, so the right is evaluated as tolerated.

7.2.2 Reasoning with role variability

In chapter 4 we introduced the algorithms that explore the variability of Role fulfillments in a Nòmos 3 model. As mentioned before, reasoning in Nòmos 3 is performed by means of backward and forward reasoning algorithms: an example of forward reasoning is the analysis of Norms applicable to Legal Roles, while an example of backward reasoning include searching for the fulfillment values of some Roles (Legal or Social).

Analysis of Norms applicable to Legal Roles. In figure 7.4 we represented the legal model extended with Legal Roles. In the example there are three Legal Roles: the U.K. custom has the right to seize the package and the duty to inspect the package (n_6, n_7); the receiver of a package has the duty to pay customs duty, VAT, and the right to collect the package (n_2, n_1, n_3); the sender has the duty to declare package content and pay excise duty (n_4, n_5). For simplicity

[11]The relationship $s_{15} \xrightarrow{\text{satisfy}} s_{12}$ is not considered as a possibility as we already know that s_{15} = SF.



UK custom
r3
receiver
r2
sender
r1
Right to seize package
n6
Duty to inspect package
n7
Right to collect package
n2
Duty to pay customs duty
n1
Duty to pay VAT
n3
Duty to pay excise duty for tobacco
n4
Duty to declare package content
n5
n5.1
reserved
holder
sat
act
block
block*
break
satisfy
satisfy*
imply
and
AU
AT
AF
T
U
F
Package is seized
s1
Inspection reveals irregularity
s2
Package goes through customs for inspection
s3
Package is collected
s4
Customs duty is payed
s5
Customs duty are irregular
s6
Value of goods is less than 135£
s7
Customs duty is less than 9£
s8
Product is declared as personal belongings in customs declaration
s9
VAT is payed
s10
Package sent from the Channel Island
s11
Package content is VAT taxable
s12
Package content is a gift worth more than 36£
s13
Package content is worth more than 15£
s14
Package content is alcohol, tobacco or fragrance
s15
Package sent from EU-special territories
s16
Package sent from non-EU country
s17
Package is posted/couriered
s18
Excise duty are payed
s19
Product is bought abroad and sent back to UK
s20
Shipped product is bought online
s21
Shipped product is received as gift
s22
Package sent from EU
s23
Package content is declared
s24
Item price >135£
s30

Figure 7.4: Example of a Nòmos 3 legal model — with Legal Roles — for tax and customs for goods sent to U.K. from abroad (https://www.gov.uk/goods-sent-from-abroad/overview): `role-fulfill-analysis()`.

we will consider the same scenario of figure 7.2 where the initial step of the algorithm has been performed, and the applicable Norms have been evaluated. Once the applicability analysis has evaluated which Norms are applicable, the algorithm 9 (`norms-applicable-lrole()`) analyzes the legal roles in the model and evaluates whether they have any Norm that applies to them.

- In the scenario the Legal Role of UK custom is holder of two Norms which are both applicable: the right n_6 to seize the package, and the duty n_7 to inspect the package. In the output of the algorithm this Legal Role has therefore two Norms applicable.

- The Legal Role of receiver is instead the holder of three Norms. The right n_2 to collect the package is applicable, the duty n_1 to pay custom duty is applicable since $s_{30}, s_{17}, s_{18} = \text{ST}$ and $s_{30} \wedge s_{17} \wedge s_{18} \xrightarrow{\text{activate}} n_1$, and also the duty n_3 to pay VAT is applicable since $s_{12}, s_{17}, s_{18} = \text{ST}$. In the output of the algorithm this Legal Role has therefore all three Norms applicable: n_1, n_2 and n_3.

- The last Legal Role of sender is holder of three Norms. The duty to pay excise duty n_4 has unknown applicability, and the duty to declare the content of the package n_5 is instead applicable. The possibility not to declare the package content $n_{5.1}$ has also unknown applicability. In the output of the algorithm this Legal Role has therefore only the Norm n_5 applicable.

7.2.3 Reasoning with compliance of a goal model

In chapter 5 we expanded the reasoning to identify also violations of Norms caused by wrong roles bringing about Situations. Reasoning over the compliance of a goal model is based both on backward and forward algorithms, as well as the identification of violation of the schema of relationships introduced in section 5.1. In the following we analyze the algorithm 5.12 to evaluate the compliance of a goal model and we focus only on the portion of the legal model that concerns the Legal Role of receiver and sender.

Goal model compliance analysis. In figure 7.1 we represented the Nòmos 3 model with both the legal and goal model for the online seller example. In the following we will consider in input to the algorithm this Nòmos 3 model, and the satisfaction value for the Situations describing the following scenario. A British online seller has sold an item for £16 ($s_{29} = \text{ST}$) and ships to the customer address in London ($s_{26}, s_{25} = \text{ST}$) the package from the warehouse in Mexico ($s_{17} = \text{ST}, s_{11}, s_{16} = \text{SF}$) together with a receipt for the purchase ($s_{33} = \text{ST}$). The customer payed for the item, shipping, and VAT ($s_{10}, s_{27}, s_{31} = \text{ST}$), and the seller has send a confirmation email ($s_{34} = \text{ST}$). The customer payed for taxes (g_{16}), however there is no indication of which tax has been payed. It is assumed that the business does not sell tobacco or alcohol ($a_1 = \text{ST}$), the customer has found the item to buy and has provided cc details ($g_{11}, g_{15} = \text{ST}$). All the other Situations in the model that have not been considered in the scenario, are considered as having unknown satisfaction value. The satisfaction value of the Goals and Situations in the goal model is highlighted in figure 7.5.

The objective of the compliance analysis algorithm (algorithm 11) is to find the compliance value of a set of Norms, given a set of Goals represented by a set of Situations with an initial satisfaction value. The current scenario represents a valid solution of the goal model: (a) the customer has achieved his goal g_{10} of buying a product and (b) the seller has achieved his goal g_{21} of managing the bought items. The U.K. address provided by the customer satisfies goal g_{13}, taxes are payed g_{16}, and the goals of paying for the item price and shipping are also satisfied. The goal g_{14} of making a payment is therefore satisfied ($g_{16} \wedge g_{17} \wedge g_{18} \xrightarrow{\text{satisfy}} g_{14}$), and together with the satisfied g_{15} they satisfy the parent goal g_{12} of making a payment. The customer's root goal of buying a product g_{10} is therefore satisfied. The seller similarly has his leaf level goals satisfied by the Situations of the scenario: payment is received (g_{27} is satisfied by g_{14}) and confirmation email is sent ($g_{24} = \text{ST}$), therefore the goal g_{24} of managing payment is achieved. Since at least the goal of preparing a receipt has been achieved, the higher goal g_{25} of preparing documents is achieved, and lastly the goal of shipping the package has also been satisfied. The seller's root goal of managing bough items is therefore satisfied (figure 7.5).

The algorithm 11 first uses forward reasoning to propagate the satisfaction values across the model (as highlighted in figure 7.6) and evaluates the compliance value of the Norms.
Starting from the right side of figure 7.6, we have that the duty n_5 to declare package content is applicable ($s_{17} \wedge s_{18} \xrightarrow{\text{activate}} n_5$, and $s_{17}, s_{18} = \text{ST}$) but however in the current scenario it is

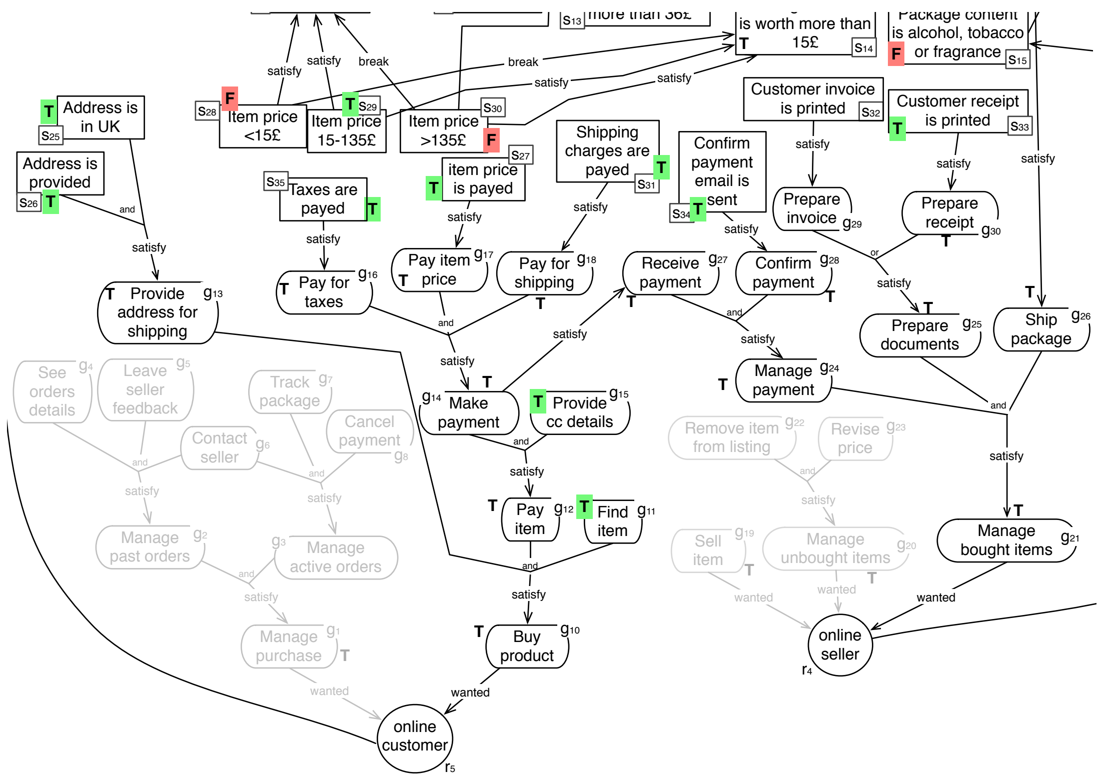


Figure 7.5: The Nòmos 3 requirements model for the online seller example, evaluated with the current scenario.

unknown whether the “package content is declared” (s_{24} = SU): the duty n_5 is applicable but has unknown satisfiability which means it is evaluated as violated. The right not to declare the content $n_{5.1}$ is inconclusive.
The duty n_4 to pay for excise duty receives false applicability ($s_{15} \xrightarrow{\text{break*}} n_4$, and s_{15} = SF) and has unknown satisfiability (s_{19} = SU): it is therefore evaluated as tolerated.
The duty n_3 to pay VAT has many incoming relationships controlling its applicability: since multiple incoming relationships to a Norm are treated as being in disjunction, it is sufficient to have one positive value to make the duty applicable (see table 3.1 in section 3.3). It is known that a package is sent (s_{18}) from a non-EU country (s_{17}) and the content of the package is VAT taxable (s_{12} = ST[12]), therefore the duty to pay VAT applies. The Situation s_{10} has however unknown satisfiability and we know if “VAT is payed”: the goal satisfied by the receiver did not specify which tax was payed. The duty n_3 is therefore evaluated as violated.

The duty n_1 to pay custom duty also has many incoming relationships controlling its applicability. While most propagate inconclusive values, the exception ($s_8 \xrightarrow{\text{block}} n_1$) is triggered: since

[12]The price of the item is 15–135£s_{29} = ST $\xrightarrow{\text{satisfy}}$ s_{14}, s_{14} = ST $\xrightarrow{\text{satisfy}}$ s_{12}.

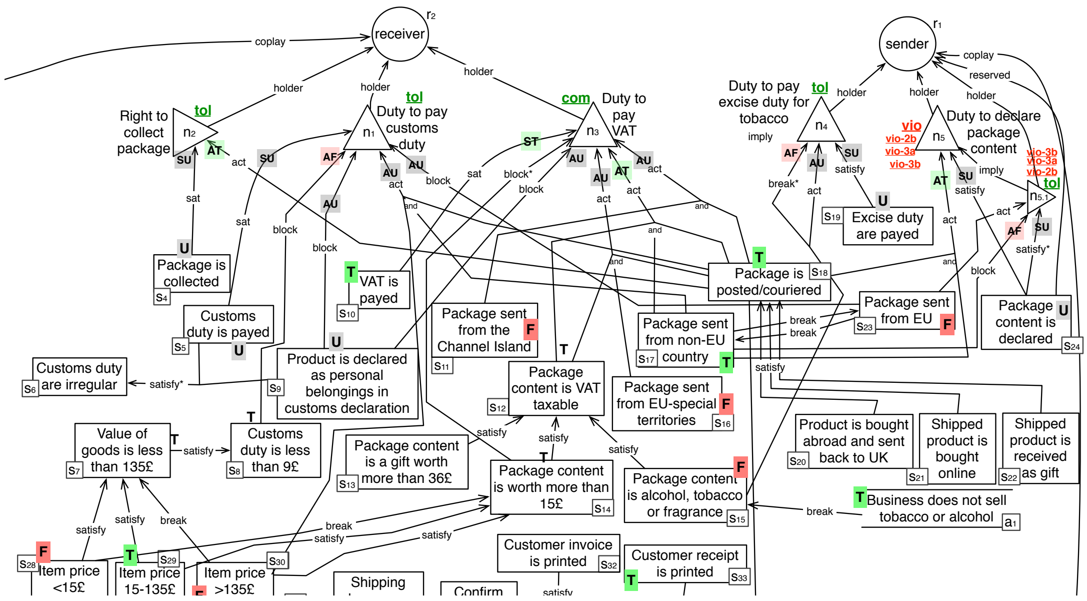


Figure 7.6: The evaluated Nòmos 3 legal model for the online seller example, evaluated with the current scenario.

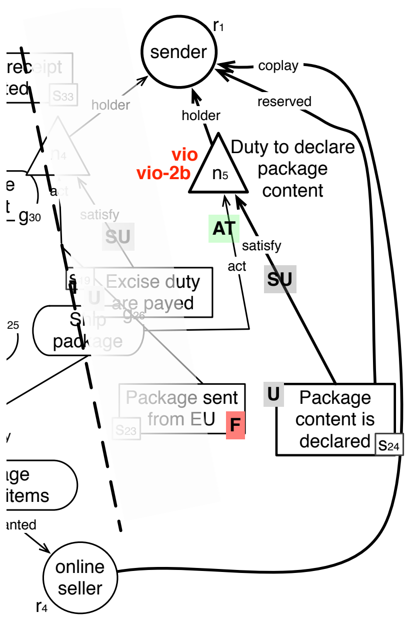


Figure 7.7: The evaluation of the reserve schema in the Nòmos 3 model for the online seller example.

custom duty is less than £9, the duty to pay custom duty is not applicable and is evaluated as tolerated. Lastly, the right to collect the package n_2 has unknown satisfiability ($s_4 = \mathrm{SU} \xrightarrow{\text{satisfy}} n_2$) but is applicable so it is evaluated as tolerated.

The last step of the algorithm is a loop evaluating whether in the model there are violations caused by the reserve schema, and update the compliance value of the Norms if needed. The reserve schema is triggered by the presence of a *reserved* relationship in the legal model. In the current example the relationship $s_{24} \xrightarrow{\text{reserved}} r_1$ in the legal model is missing its social counterpart in the requirements model (schema violation 2a, section 5.1.2): the requirements model does not specify that the Situation "Package content is declared" should be reserved for the online seller. The Norm n_5 is therefore identified as being violated and the Norm's compliance value is updated (`vio-2a`). Moreover since the Situation s_{24} is not operationalized into a goal, two more violations of the reserve schema are identified and added to the compliance values of n_5: `vio-3a` (lack of satisfy relationship from the Situation to a Goal) and `vio-3b` (lack of relationship between the goal and the social role). The same violations are identified for Norm $n_{5.1}$ which is also satisfied by the Situation reserved for the same legal role.

7.3 Conclusions

The objective of this chapter was to present our findings in the evaluation of the proposed modeling language. First we have presented an initial analysis to assess the previous modeling language (Nòmos) and evaluate the concepts that characterize the variability of the law. The results of this preliminary study have highlighted the importance of the concept of Legal Subject, while the Nòmos concept of Normative Proposition has shown some weaknesses suggesting that a different modeling approach with a finer granularity should be adopted for this atomic concept. The Nòmos 3 modeling language has then been developed to overcome some of the limitations highlighted in this analysis. The two basic concepts — Situation and Role — can be used together to form the concept of Norm, while their specializations (Legal Roles, Social Roles, Goals, . . .) allow for the representation of both the legal and requirements domain. In the second part of the chapter we have presented an illustrative case study where the proposed language is evaluated by modeling a realistic scenario. The reasoning algorithms proposed in chapters 4 and 5 are have also been evaluated on the illustrative case study.

In future work we are planning to perform an evaluation of the concepts of our modeling language to assess how well the language adapts to other laws and evaluate whether new concepts need to be included. Furthermore we are planning an evaluation of our reasoning from a legal perspective in order to evaluate our results against those of legal experts.

Chapter 8

Evaluation of the scalability

This chapter focuses on a tool support for Nòmos 3 and on the evaluation of the scalability of our approach. The research question covered is **[RQ4:]** *How well does the proposed framework performs when applied to realistic settings?*. The aim is to evaluate the robustness of our approach in real-size law.

First in section 8.1 we present our command-line tool that implements our reasoning mechanisms in terms of axioms, and uses the DLV inference engine [Alviano et al., 2010; Leone et al., 2006] to answer questions concerning compliance solutions in different situations.

Then in section 8.2 we present three experiments conducted with the tool aimed at evaluating the scalability of our legal reasoning with respect to the size of the model (measured by the number of elements in the model), the complexity of the model (measured by the inter-connectivity of elements in the model), and the space of alternatives (measured by the number of alternative refinements in a model). In order to evaluate our proposal with real-size laws, we automatically created artificial legal models of increasing sizes and with different properties which were then analyzed by the tool.

8.1 Automated reasoning

The specification and analysis of Nòmos 3 legal models is supported by a tool called Norm-Reasoning Tool, or NRTool. In this section we present this command-line reasoning tool that implements the formal semantics defined in chapter 3.3, and exploits the DLV framework to reason over the models.

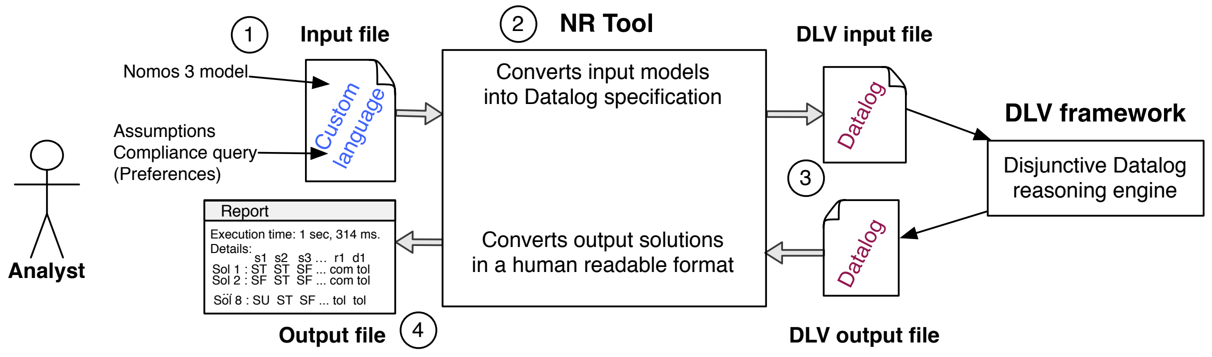
The tool takes as input a structured representation of a Nòmos 3 model, and converts it into a Datalog logic program.[1] The tool translates the model into facts and rules defining the label propagations of the language and the reserve schema. Concepts of the Nòmos 3 language are mapped into Datalog facts, while relationships are mapped into deduction rules. At the end

[1]Datalog Abiteboul et al. [1995] is a first-order logic program for querying deductive databases.

of the section we describe the Datalog rules implementing the semantics of the language, and in appendix A we have included the configuration file for the generation of the Datalog input model.

The NRTool works by translating a Nòmos 3 model into a disjunctive Datalog Abiteboul et al. [1995] program. Disjunctive Datalog is a declarative logic language and a deductive system where facts and deduction rules are expressed in the logic language. Disjunctions may appear in the rule heads to allow multiple alternative consequences to be drawn from a rule. The tool relies on DLV Alviano et al. [2010] as Datalog reasoning engine. DLV further extends disjunctive Datalog to also support weak constraints, priorities for their satisfaction, and costs for their violation. These extensions allow us to represent the preferences on the satisfaction value of pairs/group of Situations (represented as weak constraints and priorities on them), and to have an evaluation of the costs to be payed for the set of violated preferences. Concerning the search techniques and heuristics used by DLV, it implements a back search similar to SAT algorithms and advanced pruning operators, (look-ahead and look-back techniques) for model generation, and innovative techniques for answer-set checking.

The tool uses DLV to execute the logic program and perform queries on the legal model.[2] This mechanism allows to perform automated bottom-up and top-down analysis of the model, in order to support the analyst answer useful questions that about the model (chapters 4 and 5). An important characteristic of DLV is the possibility to obtain the complete set of solutions (models) produced by a set of predicates and assignments to the variables or to prune the set of models depending on the preferences and assignment specified by the decision makers.



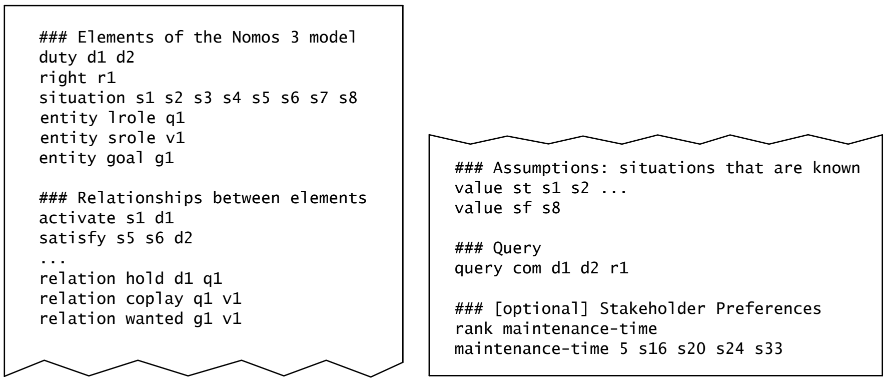
```
Execution time: 1 sec, 314 ms.
Details:
          s1  s2  s3 ...  r1  d1
 Sol 1 : ST  ST  SF ... com tol
 Sol 2 : SF  ST  SF ... com tol

 Söï 8 : SU  ST  SF ... tol  tol
```

Figure 8.1: The NRTool transforms the Nòmos 3 model provided by the analyst into a disjunctive Datalog program, and reports the output of the Datalog engine back to the analyst.

[2]DLV is an Answer Set system that extends Datalog in different ways. It adds *disjunctions* in the rule heads, thus generating multiple alternatives; it adds support for true negations; it also supports weak constraints – i.e., constraints that can be violated at a cost, allowing solutions to be ranked according to the number of violations occurring. The search techniques and heuristics used by DLV are: backward search (similar to SAT algorithms), and advanced pruning operators, for model generation and innovative techniques for answer-set checking. DLV generates as output a complete set of models produced by the set of predicates and assignments to the variables or a pruned set of models that depends on input preferences.

Figure 8.1 describes the overall behaviour of the tool.

① First, using a custom input language, the Nòmos 3 model is represented by declaring the elements and relationships of the model. In figure 8.2 we show an extract of the input file that the tool translates into Datalog and passes to the reasoner. First all the concepts of the model — Norms, Situations, Legal and Social Roles, Goals — are listed, followed by a textual description of the relationships between the elements. For example `satisfy s5 s6 d2` represents the fact that s_5 and s_6 are the source Situations of the *satisfy* relationship targeting the Norm d_2. Secondly the assumptions for the reasoning are specified: in this part of the input file the analyst can specify which Situations are known to hold/not hold in the modeled domain (both as Domain Assumption and as Goals/Situations with known satisfaction). Lastly, a *query* about the model is added, for example `query com` represents a query where the specified Norms have *com* as compliance value. Additionally, the analyst can specify as optional information a set of *preferences* between pairs or groups of Situations (as described at the bottom of figure 8.2).



```
### Elements of the Nomos 3 model
duty d1 d2
right r1
situation s1 s2 s3 s4 s5 s6 s7 s8
entity lrole q1
entity srole v1
entity goal g1

### Relationships between elements
activate s1 d1
satisfy s5 s6 d2
...
relation hold d1 q1
relation coplay q1 v1
relation wanted g1 v1
```

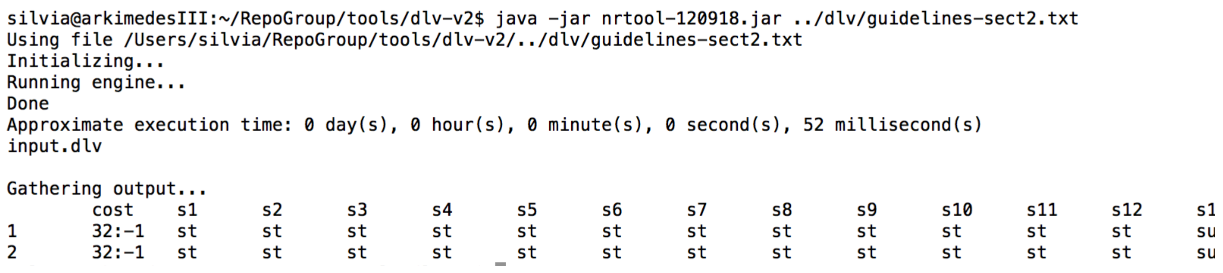
```
### Assumptions: situations that are known
value st s1 s2 ...
value sf s8

### Query
query com d1 d2 r1

### [optional] Stakeholder Preferences
rank maintenance-time
maintenance-time 5 s16 s20 s24 s33
```

Figure 8.2: An example of the input file that the NRTool translates in Datalog.

② In the second step, the tool takes the input file and translates it into a Datalog specification. All the concepts are mapped into Datalog facts representing the axioms for the concept. For example, each Situation is mapped into axioms describing its satisfiability (see code 8.1) and the satisfy relationship is mapped into the rules describing its semantics (see rule 8.15). The Datalog axioms used by the tool are detailed in the next section.

③ The tool then passes the generated Datalog representation of the model to the DLV engine which analyzes the model and returns to the tool the explored solutions to the model or the answer to the query.

④ Finally, the output of DLV is parsed by NRTool that presents the solutions to the user in the form of a report specifying the truth value of the Situations in the model and their respective compliance values. If preferences are used, then the tool returns only the best solution to the problem. In figure 8.3 we show a screenshot the output of the NRTool.



```
silvia@arkimedesIII:~/RepoGroup/tools/dlv-v2$ java -jar nrtool-120918.jar ../dlv/guidelines-sect2.txt
Using file /Users/silvia/RepoGroup/tools/dlv-v2/../dlv/guidelines-sect2.txt
Initializing...
Running engine...
Done
Approximate execution time: 0 day(s), 0 hour(s), 0 minute(s), 0 second(s), 52 millisecond(s)
input.dlv

Gathering output...
          cost    s1     s2     s3     s4     s5     s6     s7     s8     s9     s10    s11    s12    s1
1         32:-1   st     st     st     st     st     st     st     st     st     st     st     st     su
2         32:-1   st     st     st     st     st     st     st     st     st     st     st     st     su
```

Figure 8.3: An example of the output of the tool.

Datalog implementaton

In the following we introduce the Datalog rules that implement the semantics of the language used by the NRTool to translate the description of the Nòmos 3 model into a database of facts to pass to the DLV reasoning engine. The configuration file used by the tool is parametric and adjust the rules according to the multiplicity of the relationship specified in the model (it is included for reference in the appendix A).

```
1  st(s1) :- stx(s1).
2  su(s1) :- sux(s1), not stx(s1).
3  sf(s1) :- sfx(s1), not sux(s1), not stx(s1).
```

Code 8.1: Satisfiability rules for a Situation s_1: “`situation s1`” in the input file.

The satisfaction rules for a Situation (code 8.1) reflect the fact that a Situation can have three satisfaction value (ST, SU, SF). The predicate `stx(x)` is used to collect evidence about the satisfaction of a Situation, `sfx(x)` to collect evidence that a Situation is not satisfied, `sux(x)` to collect evidence on the Situation s unknown status. When there is evidence that a Situation is satisfied, the Situation is satisfied (line 1), if there is not evidence about a Situation's satisfaction it has unknown satisfaction (line 2), if there is evidence that a Situation is not satisfied, it is actually not satisfied (line 3).

```
1  st(n) :- stx(n).
2  su(n) :- sux(n), not stx(n).
3  sf(n) :- sfx(n), not sux(n), not stx(x).
4  at(n) :- atx(n).
5  au(n) :- aux(n), not atx(n).
6  af(n) :- afx(n), not aux(n), not atx(n).
```

Code 8.2: Applicability and satisfiability rules for any Norm n (both duty and right).

The applicability and satisfiability rules for a Norm work in the same way as for the satisfaction of a Situation. This code corresponds to the basic axiom for every Norm and is therefore generated for every duty or right in the model. When there is evidence of satisfiability/applicability, the Norm is satisfied/applicable (line 1 and 4), when there is not evidence about satisfaction/applicability it has unknown satisfaction/applicability (line 2 and 5), if there is evidence of false satisfiability/applicability, it is actually not satisfied/not applicable (line 3 and 6).

Similarly as before, the special predicate **stx(n)**, **atx(n)**, . . . are used to collect evidences about the satisfiability and applicability of the Norm.

```
1 com(r1) :- at(r1), st(r1).
2 tol(r1) :- at(r1), not st(r1).
3 tol(r1) :- af(r1).
4 inc(r1) :- au(r1).
```

Code 8.3: Compliance rules for a right r_1: "`right r1`" in the input file.

The compliance rules for a right reflect that a right is complied with when it is applicable and satisfied (line 1), tolerated when it is either applicable but not satisfied or when it has false applicability (line 2 and 3), and it is inconclusive when it has unknown applicability (line 4).

```
1 com(d1) :- at(d1), st(d1).
2 vio(d1) :- at(d1), not st(d1).
3 tol(d1) :- af(d1).
4 inc(d1) :- au(d1).
```

Code 8.4: Compliance rules for a duty d_1: "`duty d1`" in the input file.

The compliance rule for a duty (code 8.4) follow those of a right, with the difference that (line 2) when a duty is applicable and not satisfied it is violated.

```
1 ft(q1) :- ftx(q1).
2 fu(q1) :- fux(q1), not ftx(q1).
3 ff(q1) :- ffx(q1), not fux(q1), not ftx(q1).
```

Code 8.5: Fulfillment rules for a Social or Legal Role q_1: "`entity lrole q1`" or "`entity srole v1`" in the input file.

The rules for the fulfillment of a Legal or Social Role, follow those of satisfaction. If there is evidence that a Role is fulfilled, the Role is actually fulfilled (line 1), if there is not evidence about a Role's fulfillment it has unknown fulfillment (line 2), if there is evidence that a Role is not fulfilled, it is actually not fulfilled (line 3).

```
st(g1) :- stx(g1).
su(g1) :- sux(g1), not stx(g1).
sf(g1) :- sfx(g1), not sux(g1), not stx(g1).
```

Code 8.6: Satisfiability rules for a Goal g_1: "`entity goal g1`" in the input file.

The satisfiability rules for a Goal follow the rules of the satisfaction of a Situation (code 8.1).

The propagation rules for the relationships follow those defined in the section defining the semantics of the language (section 3.3).

```
atx(r1) :- st(s1), not st(r1s).
aux(r1) :- not st(s1), not st(r1s).
```

Code 8.7: Propagation rules for the relationship *activate* (default with sources in and-relationships): "`activate s1 r1`" in the input file.

The *activate* relationship propagates evidence of applicability to the target Norm when the source (resp. all the sources when in and-decomposition) are satisfied (line 1). Moreover for the Norm to be really activated from this relationship, the *derogate* relationship should not be propagating: the Situation `r1s` is a hidden Situation in the model used to take into account the shortcut relationship that could target the same Norm. When it is not possible to obtain evidence of positive applicability, the Norm has unknown applicability (line 2).

```
1 atx(r1) :- st(s1), not st(r1s).
2 atx(r1) :- st(s2), not st(r1s).
3 aux(r1) :- not st(s1), not st(s1), not st(r1s).
```

Code 8.8: Propagation rules for the relationship *activate* with sources in or-relationships: "`or-activate s1 s2 r1`" in the input file.

The *activate* relationship with sources in an or-relationship, propagates evidence of applicability to the target Norm when at least one of the sources is satisfied and the *derogate* relationship is not propagating (line 1–2). When it is not possible to obtain evidence of positive applicability from either sources or from *derogate* relationship, the Norm has unknown applicability (line 3).

```
1 atx(r1) :- sf(s1), sf(s2), not st(r1s).
2 aux(r1) :- not sf(s1), not sf(s2), not st(r1s).
```

Code 8.9: Propagation rules for the relationship *activate** (default with sources in and-relationships): "`activate-star s1 s2 r1`" in the input file.

The *activate** relationship propagates evidence of applicability to the target Norm when the sources are not satisfied (line 2). Moreover for the Norm to be really activated from this relationship, the *derogate* relationship should not be propagating (`r1s`). When it is not possible to obtain evidence of positive applicability, the Norm has unknown applicability (line 2).

```
1 atx(r1) :- sf(s1), not st(r1s).
2 atx(r1) :- sf(s1), not st(r1s).
3 aux(r1) :- not sf(s1), not sf(s2), not st(r1s).
```

Code 8.10: Propagation rules for the relationship *activate** with sources in or-relationships: "`or-activate-star s1 s2 r1`" in the input file.

The *activate** relationship (code **??**) with sources in an or-relationship, propagates evidence of applicability to the target Norm when at least one of the sources is not satisfied and the *derogate* relationship is not propagating (line 1–2). When it is not possible to obtain evidence of applicability from either sources or from *derogate* relationship, the Norm has unknown applicability (line 3).

```
1 afx(r1) :- st(s1), st(s2), not st(r1s).
2 aux(r1) :- not st(s1), not st(s2), not st(r1s).
```

Code 8.11: Propagation rules for the relationship *block* (default with sources in and-relationships): "`block s1 s2 r1`" in the input file.

The *block* relationship propagates evidence of negative applicability to the target Norm when the sources are satisfied (line 1). Moreover for the Norm to be really activated from this relationship, the *derogate* relationship should not be propagating (`r1s`). When it is not possible to obtain evidence of negative applicability, the Norm has unknown applicability (line 2).

```
1 afx(r1) :- st(s1), not st(r1s).
2 afx(r1) :- st(s2), not st(r1s).
3 aux(r1) :- not st(s1), not st(s2) not st(r1s).
```

Code 8.12: Propagation rules for the relationship *block* with sources in or-relationships: "`or-block s1 s2 r1`" in the input file.

The *block* relationship with sources in an or-relationship, propagates evidence of negative applicability to the target Norm when at least one of the sources is satisfied and the *derogate* relationship is not propagating (line 1–2). When it is not possible to obtain evidence of negative applicability from either sources or from *derogate* relationship, the Norm has unknown applicability (line 3).

```
1 afx(r1) :- sf(s1), sf(s2), not st(r1s).
2 aux(r1) :- not sf(s1), not sf(s2), not st(r1s).
```

Code 8.13: Propagation rules for the relationship *block** (default with sources in and-relationships): "`block-star s1 s2 r1`" in the input file.

The *block** relationship propagates evidence of negative applicability to the target Norm when the sources are not satisfied (line 1). Moreover for the Norm to be really activated from this relationship, the *derogate* relationship should not be propagating (`r1s`). When it is not possible to obtain evidence of negative applicability, the Norm has unknown applicability (line 2).

```
1 afx(r1) :- sf(s1), not st(r1s).
2 afx(r1) :- sf(s2), not st(r1s).
3 aux(r1) :- not sf(s1), not sf(s2), not st(r1s).
```

Code 8.14: Propagation rules for the relationship *block** with sources in or-relationships: "`or-block-star s1 s2 r1`" in the input file.

The *block** relationship with sources in an or-relationship, propagates evidence of negative applicability to the target Norm when at least one of the sources is not satisfied and the *derogate* relationship is not propagating (line 1–2). When it is not possible to obtain evidence of negative applicability from either sources or from *derogate* relationship, the Norm has unknown applicability (line 3).

```
1 stx(r1) :- st(s1).
2 sux(r1) :- not st(s1).
3 satisfy(s1,r1).
```

Code 8.15: Propagation rules for the relationship *satisfy* (default with sources in and-relationships): "`satisfy s1 r1`" in the input file.

The *satisfy* relationship propagates evidence of satisfiability to the target Norm when the source (resp. all the sources when in and-decomposition) are satisfied (line 1). When it is not possible to obtain evidence of positive satisfiability, the Norm has unknown satisfiability (line 2). The last line is a fact about the relationship used for the evaluation of the reserve schema.

```
1 stx(r1) :- st(s1).
2 stx(r1) :- st(s2).
3 sux(r1) :- not st(s1), not st(s2).
4 satisfy(s1,r1).
5 satisfy(s2,r1).
```

Code 8.16: Propagation rules for the relationship *satisfy* with sources in or-relationships: “`or-satisfy s1 s2 r1`” in the input file.

The *satisfy* relationship with sources in an or-relationship, propagates evidence of satisfiability to the target Norm when at least one of the sources is satisfied (line 1–2). When it is not possible to obtain evidence of positive satisfiability from either sources, the Norm has unknown satisfiability (line 3). The last two lines are facts used for the evaluation of the reserve schema.

```
1 stx(r1) :- sf(s1), sf(s2).
2 sux(r1) :- not sf(s1), not sf(s2), not st(r1s).
3 satisfy(s1,r1).
4 satisfy(s2,r1).
```

Code 8.17: Propagation rules for the relationship *satisfy** (default with sources in and-relationships): “`satisfy-star s1 s2 r1`” in the input file.

The *satisfy** relationship propagates evidence of satisfiability to the target Norm when all the sources in an and-decomposition are not satisfied (line 1). When it is not possible to obtain evidence of positive satisfiability, the Norm has unknown satisfiability (line 2). The last lines are facts used for the evaluation of the reserve schema.

```
1 stx(r1) :- sf(s1).
2 stx(r1) :- sf(s2).
3 sux(r1) :- not sf(s1), not sf(s1).
4 satisfy(s1,r1).
5 satisfy(s2,r1).
```

Code 8.18: Propagation rules for the relationship *satisfy** with sources in or-relationships: “`or-satisfy-star s1 s2 r1`” in the input file.

The *satisfy** relationship with sources in an or-relationship, propagates evidence of satisfiability to the target Norm when at least one of the sources is not satisfied. When it is not possible to obtain evidence of positive satisfiability from either sources, the Norm has unknown satisfiability (line 3). The last two lines are facts used for the evaluation of the reserve schema.

```
1 sfx(r1) :- st(s1), st(s2).
2 sux(r1) :- not st(s1), not st(s2).
```

Code 8.19: Propagation rules for the relationship *break* (default with sources in and-relationships): “`break s1 s2 r1`” in the input file.

The *break* relationship propagates evidence of negative satisfiability to the target Norm when all the sources in and-decomposition are satisfied (line 1). When it is not possible to obtain evidence of negative satisfiability of the Norm, the Norm has unknown satisfiability (line 2).

```
1  sfx(r1) :- st(s1).
2  sfx(r1) :- st(s2).
3  sux(r1) :- not st(s1), not st(s2).
```

Code 8.20: Propagation rules for the relationship *break* with sources in or-relationships: “`or-break s1 s2 r1`” in the input file.

The *break* relationship with sources in an or-relationship, propagates evidence of negative satisfiability to the target Norm when at least one of the sources is satisfied (line 1–2). When it is not possible to obtain evidence of negative satisfiability from either sources, the Norm has unknown satisfiability (line 3).

```
1  sfx(r1) :- sf(s1), sf(s2).
2  sux(r1) :- not sf(s1), not sf(s2).
```

Code 8.21: Propagation rules for the relationship *break** (default with sources in and-relationships): “`relation break-star s1 s2 r1`” in the input file.

The *break** relationship propagates evidence of negative satisfiability to the target Norm when all the sources in and-decomposition are not satisfied (line 1). When it is not possible to obtain evidence of negative satisfiability of the Norm, the Norm has unknown satisfiability (line 2).

```
1  sfx(r1) :- sf(s1).
2  sfx(r1) :- sf(s2).
3  sux(r1) :- not sf(s1), not sf(s2).
```

Code 8.22: Propagation rules for the relationship *break** with sources in or-relationships: “`relation or-break-star s1 s2 r1`” in the input file.

The *break** relationship with sources in an or-relationship, propagates evidence of negative satisfiability to the target Norm when at least one of the sources is not satisfied. When it is not possible to obtain evidence of negative satisfiability from either sources, the Norm has unknown satisfiability (line 3).

```
1  vio(r1) :- vio(r2).
2  inc(r1) :- inc(r2), not vio(r2).
3  tol(r1) :- tol(r2), not inc(r2), not vio(r2).
4  com(r1) :- com(r2), not tol(r2), not inc(r2), not vio(r2).
```

Code 8.23: Propagation rules for the relationship *imply*: “`imply r2 r1`” in the input file.

The *imply* relationship is used to propagate evidence of compliance between two Norms. So when the source Norm is violated, the target Norm is also violated (line 1). If it is not violated but it is inconclusive, then it is inconclusive (line 2). Otherwise the Norm is tolerated (line 2) or complied with (line 4).

```
1 st(r1s) :- stx(r1s).
2 su(r1s) :- sux(r1s), not stx(r1s).
3 sf(r1s) :- sfx(r1s), not sux(r1s), not stx(r1s).
4 stx(r1s) :- com(r2).
5 sux(r1s) :- not com(r2).
6 afx(r1) :- st(r1s).
7 aux(r1) :- not st(r1s).
```

Code 8.24: Propagation rules for the relationship *derogate*: “`derogate r2 r1`” in the input file.

The *derogate* relationship is used to propagate to the target Norm negative applicability when the source Norm is complied. First the hidden Situation `r1s` for the target Norm is created (line 1–3). Then if the source Norm is complied with `com(r2)`, then there is evidence that the Situation `r1s` is satisfied (line 4), otherwise the Situation has unknown satisfiability (line 5). When the hidden Situation `r1s` is satisfied `st(rs1)`, then there is evidence that the Norm has negative applicability (line 6), otherwise it has unknown applicability (line 7).

```
1 st(r1e) :- stx(r1e).
2 su(r1e) :- sux(r1e), not stx(r1e).
3 sf(r1e) :- sfx(r1e), not sux(r1e), not stx(r1e).
4 st(r1e) :- com(r2).
5 su(r1e) :- not st(r1e).
6 atx(r1) :- st(r1e).
7 aux(r1) :- not st(r1e).
```

Code 8.25: Propagation rules for the relationship *endorse*: “`endorse r2 r1`” in the input file.

The *endorse* relationship is used to propagate to the target Norm positive applicability when the source Norm is complied. As before, the hidden Situation `r1e` for the target Norm is first created (line 1–3). Then if the source Norm is complied with, then there is evidence that the Situation `r1e` is satisfied (line 4), otherwise the Situation has unknown satisfiability (line 5). When the hidden Situation `r1e` is satisfied, then there is evidence that the Norm has positive applicability, otherwise it has unknown applicability (lines 6 and 7).

```
1 stx(q1r1) :- com(r1).
2 stx(q1r1) :- tol(r1).
3 ftx(q1) :- stx(q1r1).
4 ffx(q1) :- vio(r1).
5 fux(q1) :- au(r1).
6 holder(r1,q1).
```

Code 8.26: Propagation rules for the relationship *holder*: “`relation holder r1 q1`” in the input file.

The *holder* relationship is used to propagate fulfillment from a Norm to a Legal Role. The semantics of the relationship is such that when the Norm is complied with or tolerated, the Legal Role is fulfilled. These two options are implemented as different ways for satisfying the hidden Situation `q1r1` (line 1 and 2): so when the hidden Situation is satisfied, there is evidence that the Legal Role is fulfilled (line 3). The Legal Role is not fulfilled only if the Norm is violated (line 4), otherwise it has unknown fulfillment. The last line is used for the evaluation of the reserve schema.

```
reserved(s1,q1).
```

Code 8.27: Propagation rules for the relationship *reserved*: "`relation reserved s1 q1`" in the input file.

The *reserved* relationship does not propagate any values, and it is only a fact used for the evaluation of the reserve schema.

```
1 ftx(v1) :- st(g1), ftx(v1cop).
2 fux(v1) :- not ftx(v1), fux(v1cop).
3 ffx(v1) :- not ftx(v1), not fux(v1cop).
4 wanted(g1,v1).
```

Code 8.28: Propagation rules for the relationship *wanted*: "`relation wanted g1 v1`" in the input file.

The *wanted* relationship is used to propagate fulfillment from a Goal to a Social Role. When the Goal is satisfied and (all) the coplayed Legal Roles are fulfilled (only one in this example, `ftx(v1cop)`), then the Social Role is fulfilled. Otherwise, if the Goal is not satisfied and the Legal Role has unknown/negative fulfillment, the Social Role has unknown/negative fulfillment (line 2, 3). The last line is used for the evaluation of the reserve schema.

```
1 ftx(v1cop) :- ftx(q1).
2 ffx(v1cop) :- ffx(q1).
3 fux(v1cop) :- not ftx(v1cop), not ffx(v1cop).
4 coplay(q1,v1).
```

Code 8.29: Propagation rules for the relationship *coplay*: "`relation coplay q1 v1`" in the input file.

If there is evidence that the Legal Role is fulfilled/not-fulfilled, then there is evidence that for that *coplay* relationship the Social Role is fulfilled/not-fulfilled (resp. line 1 and 2). Otherwise there will be unknown evidence about fulfillment (line 3). The last line is a predicate about the relationship that is used for the reserve schema.

```
schematwo(q1,v1,s1) :- coplay(q1,v1), reserved(s1,q1), reserved(s1,v1).
schematwoa(q1,v1,s1) :- reserved(s1,q1), reserved(s1,v1).
schematwob(q1,v1,s1) :- coplay(q1,v1), reserved(s1,q1).
viotwoa(r1) :- not schematwo(q1, v1, s1), schematwoa(q1,v1,s1).
viotwob(r1) :- not schematwo(q1, v1, s1), schematwob(q1,v1,s1).
schemathree(s1,g1,v1) :-  reserved(s1,v1), wanted(g1,v1), satisfy(s1,g1).
```

```
 7  schemathreea(s1,g1,v1) :-  reserved(s1,v1), wanted(g1,v1).
 8  schemathreeb(s1,g1,v1) :-  reserved(s1,v1), satisfy(s1,g1).
 9  viothreea(r1) :- not schemathree(s1,g1,v1), schemathreea(s1,g1,v1).
10  viothreeb(r1) :- not schemathree(s1,g1,v1), schemathreeb(s1,g1,v1).
```

Code 8.30: Compliance rules for the reserve schema.

In code 8.30 we have encoded the compliance rules for the reserve schema, which are generated when there is a *reserved* relationship from a Situation to a Legal Role. The schema is actually decomposed in the two parts.

1. First the predicate `schematwo` is used to correctly identify the part of the reserve between the Situation, Legal and Social Role (line 1). Then the two predicates `schematwoa` and `schematwob` are used to identify a partial reserve schema. If the predicate for the correct schema does not hold and the predicate `schematwoa` holds (line 4), then the Norm[3] is violated with label `vio-2a` (predicate `viotwoa`). Similarly the violation `vio-2b` of the Norm is identified if the predicate for the correct schema does not hold and the predicate `schematwob` holds (line 5).

2. The predicate `schemathree` is used to correctly identify the part of the reserve between the Situation, Goal and Social Role (line 6). As before two predicates `schemathreea` and `schemathreen` are used to identify this part of the reserve schema when a relationship is missing. If the predicate for the correct schema does not hold and the predicate `schemathreea` holds (line 7), then the Norm is violated with label `vio-3a` (predicate `viothreea`). Similarly the violation `vio-3b` of the Norm is identified if the predicate for the correct schema does not hold and the predicate `schemathreeb` holds (line 8).

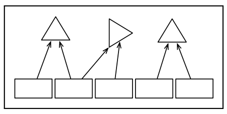
8.2 Scalability of the approach

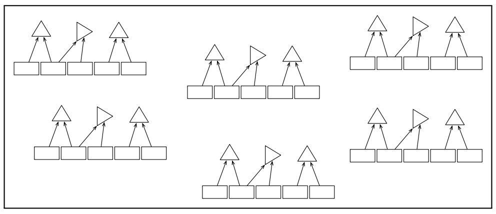
Laws usually consist of tens or even hundreds of pages of natural language text, resulting in large models that may involve tens of thousands of concepts and links. In this section we investigate the scalability of our reasoning tool with respect to three criteria: the size of the legal model, the complexity of the legal model, and the space of alternatives in the model. The experiments were performed in the first version of the language called Nòmos 2, which represents the concepts and relationships in the legal model (Norms and Situations) without the Legal Roles.

In order to investigate these three scalability directions we have set up a testing framework, capable of producing artificial Nòmos 2 models with desired properties, run compliance analysis and record execution times. The NRTool was adapted in order to still accept as input a manually created model (figure 8.4a), then duplicate this input model (figure 8.4b), and link these duplicated models in order to reach a single correct legal model (figure 8.4c). The duplication and

[3]The norm that the Legal Role q_1 holds and that it is satisfied by s_1.

connection of models was controlled according a to different parameters: number of duplication of the seed model, number of relationship connecting the duplicated, number of constraints to add, and other useful parameters.


(a) The first step is the creation of the input model.


(b) The second step is the duplication of the input model.

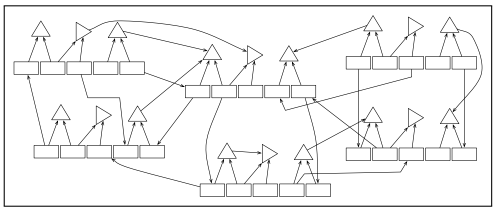

(c) The third step is the connection of the models with random relationships.

Figure 8.4: The three steps used by the generation-algorithm to generate artificial Nòmos 2 legal models.

8.2.1 Design of the experiment

Research questions. The objective of this section is to analyze the scalability of our legal reasoning NRTool with respect to the following three research questions:

RQ1 How does the tool scale with respect to the size of the problem, defined as the number of elements in the legal model?

RQ2 How does the tool scale with respect to the complexity of the problem, defined as the number of relationships constraining the different elements of the legal model? Also, how does the tool scale w.r.t. the number of solutions?

RQ3 How does the tool scale with respect to the space of alternatives, defined as the number of alternative refinements?

Experiment setup. In order to investigate these research questions we have performed three experiments:

1. To answer the first research question we have set up an experiment that tests the behaviour of the tool as the size of the Nòmos 2 model increases. The size of the model is measured in number of total concepts, (i.e., nodes in the graph representing the model). A first model (the *input model*) was initially manually created, consisting of 4 norms and 10 situations. Starting from this input model, 50 test models were then automatically generated with a number (from 1 to 50) of replicas of the input model, resulting in models of increasing size from 15 to 13875 nodes. To ensure that our models had a sufficient connectivity, a fixed number of 10 random relationships was added between replicas. The randomness of

these relationships was controlled by a parameter in our configuration called 'seed'.[4] The experiment was run 5 times with the same input model but different seeds.

2. Answering the second research question requires understanding how the tool behaves when the connectivity of its input model changes. The complexity of the problem was measured with the number of relationships added between replicas. As in the previous case, we started from an input model and produced 700 test models. Differently from the previous case, the size of the model was kept fixed in this experiment while the number of relationships added between the replicated model was increased to evaluate its impact in the performance of the tool. In order to minimize the impact of the size of the model in the evaluation, the final model was kept to a size of 225 nodes. Relationships were then gradually added from 0 to 750 to produce the test model.[5]

3. To answer the last research question we set up an experiment to analyze how to tool behaved as the number of solutions increased. In order to evaluate the impact of the number of solution, we studied the impact of specific constructs that magnify the number of available solutions in the model: and- and or-relationships. In this experiment the size of the model and its connectivity were fixed, while the variability was manually and structurally introduced in the different input models. In order to magnify the effects of the variability that is under analysis, the size of the model was very large (14000 nodes). 5 variations of the same input model have been created where the proportion between AND- and OR-relationships was incrementally inverted: from a value of OR of 0% (i.e., all the relationships are in AND) to 100% (i.e. all the relationships are in OR).

Measures and equipment. In all three experiments the measure for the evaluation of the tool is the time it took to the reasoning tool to explore the solution space and return the solutions. All experiments have been performed on an Intel i7 eight core 2.80 GHz computer equipped with 6 GB of memory running Linux version 2.6.18. The tool, the setting data, and the results generated by the experiments are available at `http://selab.fbk.eu/lawvariability/`.

8.2.2 Experiment results

RQ1. The results of this experiment are reported in figure 8.5. The figure reports on the x-axis the size of the model expressed as number of concepts of the test model. The y-axis reports the time taken at each execution to identify the set of solutions. As we can see all five runs have a linear trend, therefore guaranteeing that different random seeds used to variate the type of relationships used to connect the duplicated models have no significant impact in the overall type of trend. The size of the model reached 14000 elements and, as a comparison, we estimated

[4]By changing this seed, the random relationships also change, thus creating similar but not identical test models.

[5]The type of relationships added was random and controlled by the 'seed' parameter.

HIPAA's section 164.502 to be of around 4000 Situations, well within the boundaries of the sizes of the models tested. On average the tool took around 2.7 seconds to return the solutions in the largest model.

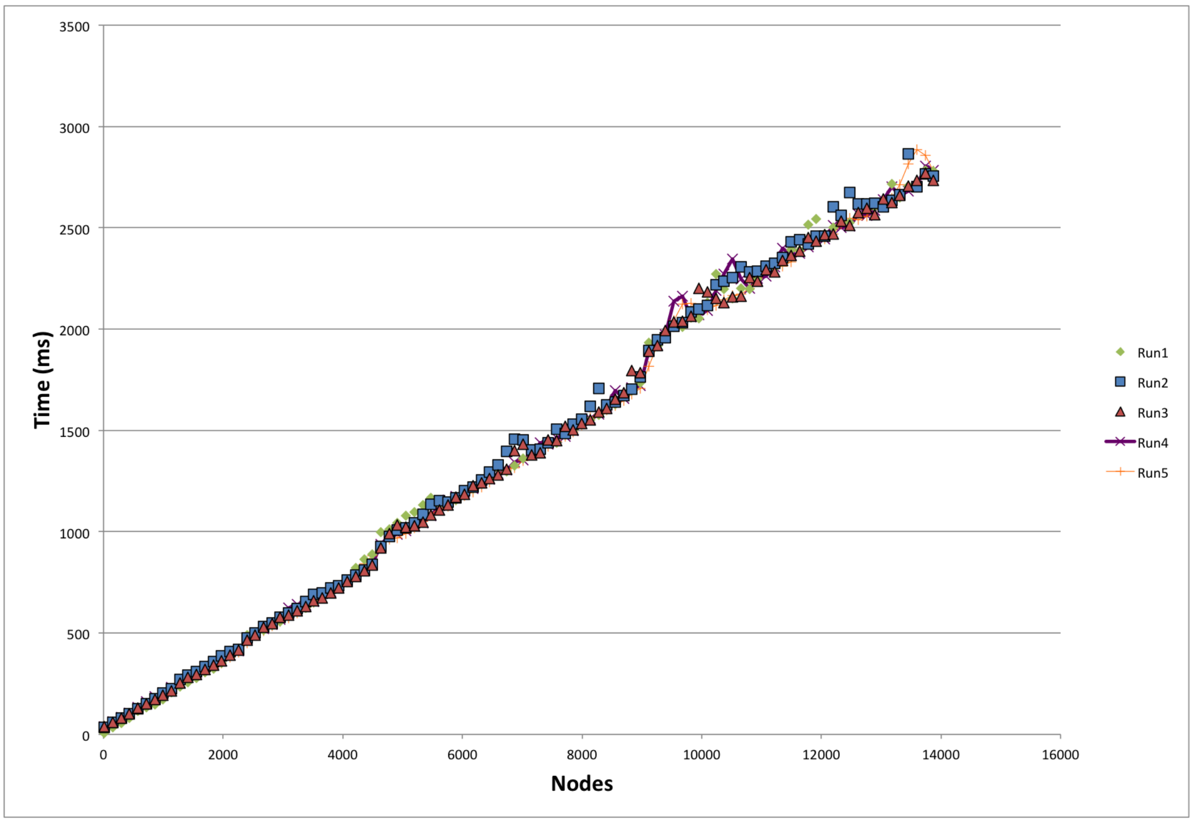


Figure 8.5: Results from the first scalability experiments analyzing the size of the problem.

RQ2. The results of the second experiment are reported in figure 8.6. In this case the x-axis reports the "connectivity" parameter — i.e., how many random relationships have been added to the model. As before the y-axis reports the time taken at each execution to find the solutions. The left figure reports the execution time for all the connectivity values. The first 7 runs reached the timeout of 60 seconds: these runs are indeed identifiable as the initial outliers at the beginning of the plot. From this initial plot that included the outliers was not possible to evaluate a trend in the remaining execution time which was significantly lower. To this end, we magnified in the right figure the runs from a value of connectivity 8 to 500 and basically highlight the trend that is otherwise not possible to see in the left figure. The execution time for these runs ranged between 35ms and 70ms: as we can see in the right side of figure 8.6 the execution time first decreases slightly until a connectivity of approximately 50 is reached, and then slowly increases again. The reason of this behavior is that for unconstrained Nòmos 2 models (i.e., models with few relationships among nodes) the number solutions depends exponentially on the number of nodes N (3^N, to be exact). As relationships are added, the number of solution decreases, as does the time to find all of them. As more relationships are added, the complexity of the problem to be solved — defined by the number of relationships over a fixed graph — overtakes the cost of finding all solutions. This peculiarity results in the increasing trend shown on the right plot of figure 8.6.

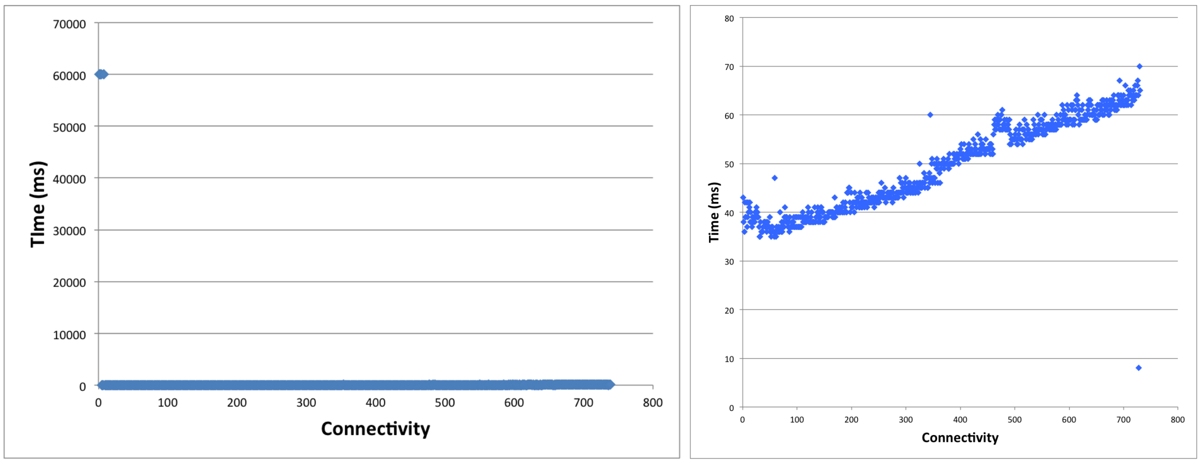


Figure 8.6: Results from the second scalability experiments analyzing the complexity of the problem.

RQ3. The results of the experiment for the last research question are shown in figure 8.7. The plot shows the five input model and the results for each of the ten runs of the experiment. The results show that on average each input model was executed within the same time range over the 10 runs, only one outlier data was noticed in the fourth run of the last input model. As we can see from figure 8.7 the input models with low connectivity values of or-0, or-25 and or-50 — respectively where there was 0%, 25%, and 50% or-relationships — have roughly the same results with an average execution time of 230ms. The results from the input models with connectivity value of or-75 and or-100 where the or-decomposition becomes prevalent, times increase by approximately 20% passing from an average of 230 ms to 275 ms.

8.2.3 Discussion and limitations

The results of these experiments are twofold. On the one hand, we see a very encouraging linear trend for execution times, which generally corresponds to at most a few seconds. On the other hand, in the second experiment, we see in some cases times running out of bounds. This is due to the difference between *searching for a solution set* and *exploring the solution set*. The exploration time may overcome search time and diverge if the model is highly sparse and the space of alternatives is extremely large. Also with the third experiments we confirmed how the space of alternatives directly influences execution time. Moreover, as we can see from the second

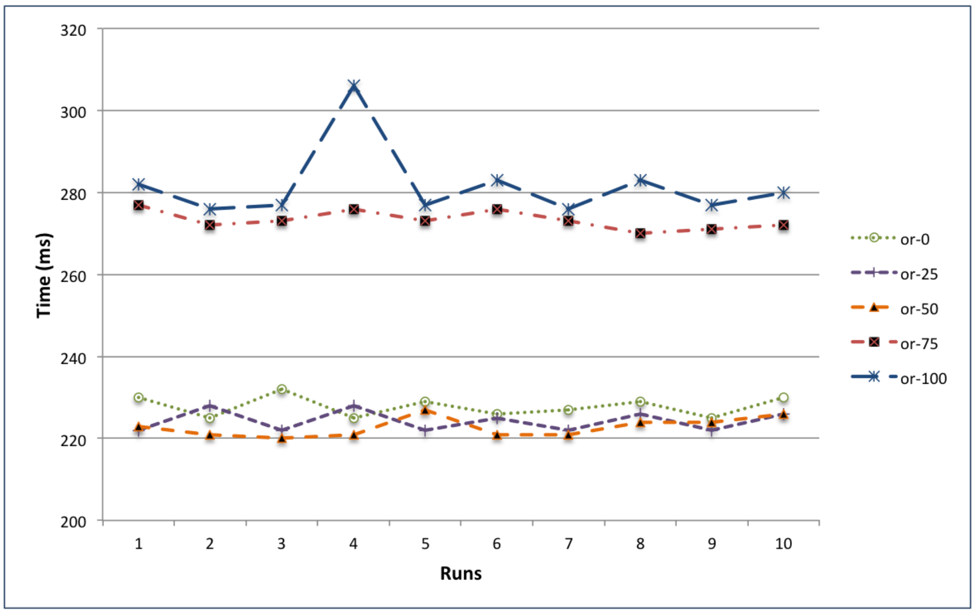


Figure 8.7: Results from the third scalability experiment analyzing the space of alternatives.

experiment, the initial constraints added to the model resulted first in a reduction of the time (as the number of solutions was decreasing) but then complexity kicks-in increasing overall execution times. The results of these experiments are comparable with similar investigations performed (see for example [Patel-Schneider and Sebastiani, 2003]). The lesson learnt from these experiments is that conceptual models can be a viable solution in analyzing laws for compliance, but only if the modelled laws are not too under-constrained. Given that laws are generally comprised of a high number of conditions, exceptions, derogations, cross-references and so on, we do not expect that real law models are under-constrained.

8.3 Conclusions

The objective of this chapter was to present an evaluation of the scalability of our approach and reasoning. In order to deal with the problem of regulatory compliance, it is important to perform analysis and reasoning of the legal models. Laws can however be very long and complicated documents — therefore suggesting large and complex models — so we performed an evaluation aimed at assessing the scalability of our proposal.

First we have presented our reasoning tool that implements the semantics of the Nòmos 3 language (see chapter 3). The command-line tool takes as input a description of the Nòmos 3 model, and converts it into Datalog facts and rules. The tool then exploits the DLV engine to explore the space of alternatives and to return the solution to a given query. In the second part of the chapter we have presented the results of three experiments that used the tool in order to evaluate the scalability of our approach with the use of large artificial models of laws. Our results show that the tool scales to problems of moderate law size and of moderate complexity. Our findings also confirm the space of alternatives has indeed impact on the performance of the reasoning which are worsen of approximately 20%.

Chapter 9

Evaluation of the process

In this chapter we cover the research question **[RQ4:]** *How well does the proposed framework performs when applied to realistic settings?*. The aim is to perform an evaluation of the systematic process proposed in chapter 6.

In the first part of the chapter (section 9.1) we will show the results of an initial analysis aimed at evaluating the initial version of the systematic process. The context of the evaluation was an Italian organization involved in the design and development of a project called CSS "Cartella Socio Sanataria" (*Electronic Health and Social Record*) aimed at monitoring healthcare and social processes.

After presenting the findings of this preliminary evaluation, we will describe in the second part of the chapter (section 9.2) the evaluation of the process with the use of an illustrative case study. In this section, we will use the same scenario of chapter 7, and will illustrate our process in the context of a software to help a British seller manage taxes and paperwork for shipping goods to U.K.

9.1 Initial evaluation of the systematic process

At the early stage of the research we performed a preliminary empirical evaluation, based on an industrial case study, with the aim of evaluating the adequacy of the initial version of the proposed systematic process to revise a set of requirements to comply with applicable regulations. The object of the case study was the design of a system for managing Italian Electronic Health Record (EHR) data. Such systems need to comply with existing laws in force (e.g. Italian Privacy Law, and Guidelines on EHR[1]).

Like our proposed process (see chapter 6), also the initial version of the process was composed of three phases: an analysis phase, a compliance check, and a revision phase that looped back to the analysis phase. Since the earlier version of the process was based on Nòmos [Siena, 2010] (see

[1] `http://www.garanteprivacy.it/web/guest/home/docweb/-/docweb-display/docweb/1672821`.

section 2.1.1) extended with the concept of irregularities [Ingolfo et al., 2012] (i.e., violations), each phase was divided in different steps evaluating different aspects of the model according to the characteristic of that framework. The two main differences between the initial version of the processes and the one proposed in chapter 6 are that (1) the earlier version evaluated revisions and amendments to the model based on an argumentation framework, and (2) that had no reasoning mechanism underlying the language.

The context. The context of the study was an Italian organization involved in the design and development of a project called CSS “Cartella Socio Sanataria" (*Electronic Health and Social Record*) aimed at monitoring healthcare and social processes in Trentino, a region in northern Italy. The main goal of the project was to support information sharing among different health care entities, such as hospitals, family doctors, and other agencies (for social, mental health and other medical services). Sharing information among CSS participants is regulated by Privacy Laws, so the system needed to be designed and created according to the principles set forth by the Italian Privacy Law and the Guidelines on EHR. The context of our case study, is therefore the initial requirements model of the CSS project that needs to be designed to address the stakeholder needs and tailored to comply with these regulations.

9.1.1 Design

Research questions. The goal of this initial study was twofold: (a) to evaluate the conceptual adequacy of the proposed framework and (b) the adequacy of the revision process in the context of the CSS project that has to comply with Italian guidelines on EHR and Privacy Laws. In this section we will focus on the second objective and its research question: *Is the proposed process for establishing compliance appropriate for the problem-at-hand?*

Measures. We adopted two measures. The first measure was the uncertainty of the stakeholders in performing the tasks related to each step of the process. This measure was obtained from the session recordings and transcripts (our first source of evidence). We used heuristics in order to identify manifestations of uncertainties such as long pauses and questions arisen. To evaluate the adequacy of the process, we looked for manifestations where the participants expressed their concerns with respect to the objective of the step or voiced problems with respect to the logic used in the process.

The second measure was aimed at measuring the adequacy of the process that was perceived by the participants and was obtained from a post-study interview of the participants (our second source of evidence). To this end, the participants were asked to qualitatively evaluate (0–5 scale) the process and its helpfulness. We investigated if and where the participants felt the need of more systematic support from the process, and where they perceived that the process was not supportive enough.

Materials and participants. In our pilot study we have considered the compliance of the goal-model of the CSS project with respect to two legal extracts: one from the Guidelines on EHR concerning the right of a patient to cancel specific information from the EHR (often referred to as "blanking right"), and the other one from the Data Privacy code concerning the patient's right to obtain the list of subject(s) authorized to receive his/her personal information (Section 3 comma 11, and Section 7 comma 2.e).

The participants in this pilot case study were chosen to reflect the strategic nature of a compliance discussion that was at the basis of the argumentation framework. Two domain experts — active in the development of CSS and directly involved with the stakeholders — participated in representing stakeholder needs and views. Two technical experts with good experience in goal-modeling acted as requirements engineers. Two legal experts — a lawyer specialized in security and privacy issues, and another expert in data privacy — were involved in providing feedback with respect to legal issues. Lastly, a mediator was responsible for guiding the process through the steps, and for managing the discussions targeted at the revision of the model.

9.1.2 Execution of the case study

Our pilot case study was conducted with a series of three 2-hour meetings aimed at reproducing the steps of the initial proposal of the systematic proposed. Before the actual study, we performed a pre-pilot meeting to test the setting of our study. During this initial assessment we performed a dry-run discussion, and evaluated the methods to share the materials during the meetings. At the beginning of every meeting we performed a brief recap of the process and outlined the agenda for the day. After this briefing, the participants were guided through the steps of the process in order to evaluate and revise the goal model of the requirements. All sessions have been video recorded in order to access later on to the discussions and their transcripts for their analysis. Individual interviews were performed after the study was completed to asses the views of the participants with respect to our two research questions.

9.1.3 Results

	TRANSCRIPT	INTERVIEW	
	# manifestations of uncertainty	average difficulty (0–5, easy–very hard)	standard deviation
Process	0	4.4	0.4

Table 9.1: The numerical results of the pilot case study. The adequacy of the framework concepts and process has been validated against two measures: one from the transcripts of the session recordings, and one from post-study interview of the participants.

Is the proposed process for establishing compliance appropriate for the problem-at-hand? During the execution of case study, the role of mediator acted as interface for the process and guided the participants through the three phases and corresponding steps. While there were no situations where the mediator (following the process) interfered with the study, in a few cases we noticed that more support would have been useful. In fact, choosing the tactic for the exploration and analysis of the goal-model (e.g. using a depth-first or breadth-first strategy) was a task not supported in the process or framework, thereby allowing the discussants to choose a strategy on a case-by-case basis. Analyzing the session recordings, we noticed for example that top level goals were mainly analyzed using a breadth-first approach, while lower level goals were investigated with a depth-first approach. From this analysis we were also able to identify other possible optimizations of the process. For example, during the first step of the analysis phase the participants had to consider all possible combinations of legal subjects and actors (which in our later version of the process would be equivalent to evaluating the link between legal and social roles). In this context the participants were asked to argument also cases where the relationship was obviously there (or not there). The argumentative framework used required arguments supporting the existence/absence of this relationship, but the participants found that few arguments were possible. Moreover the legal experts were far more decisive in this step than the stakeholders or the technical experts.

Other optimizations to the process were found to be related to the lack focus and continuity of the analysis-revision phase. In this earlier version of the process *all* possible irregularities in the model were discussed,[2] in a later moment all the solutions were discussed, and finally all the solutions were evaluated. We believe this step would be more efficient and simple if one irregularity and its solution were discussed at the time, in order not to lose focus on the issue at hand. Despite all these important optimization, we never found in the transcripts instances were the purpose of the steps or their helpfulness was doubted. In general, we can say that the process was found suitable and adequate to handle the problem.

The results of the analysis of the interviews are in line with the findings of earlier observations, and are reported in table 9.1. None of the participants found that the process was unhelpful, or interfered with design activities, and they all expressed positive opinions on the process. Stakeholders valued the fact that the process followed the activities and discussions naturally occurring during software development, and also felt that this direct confrontation among techcnical-legal-stakeholders was useful in (discovering and) clarifying ambiguities. In fact — even though the participants noted some of the optimization issues discussed above — their general opinion was that the process was extremely appropriate to handle the problem. A common comment was made on the usefulness of the mediator. Participants, for example, worried that discussion on a topic may diverge, and they felt that this role may be critical in avoiding/limiting the problem.

[2]An irregularity is defined as a violation in the requirements model where a Goal is in violation with a given norm. In order to establish this irregularity, all combinations of goals-norms in the model were discussed.

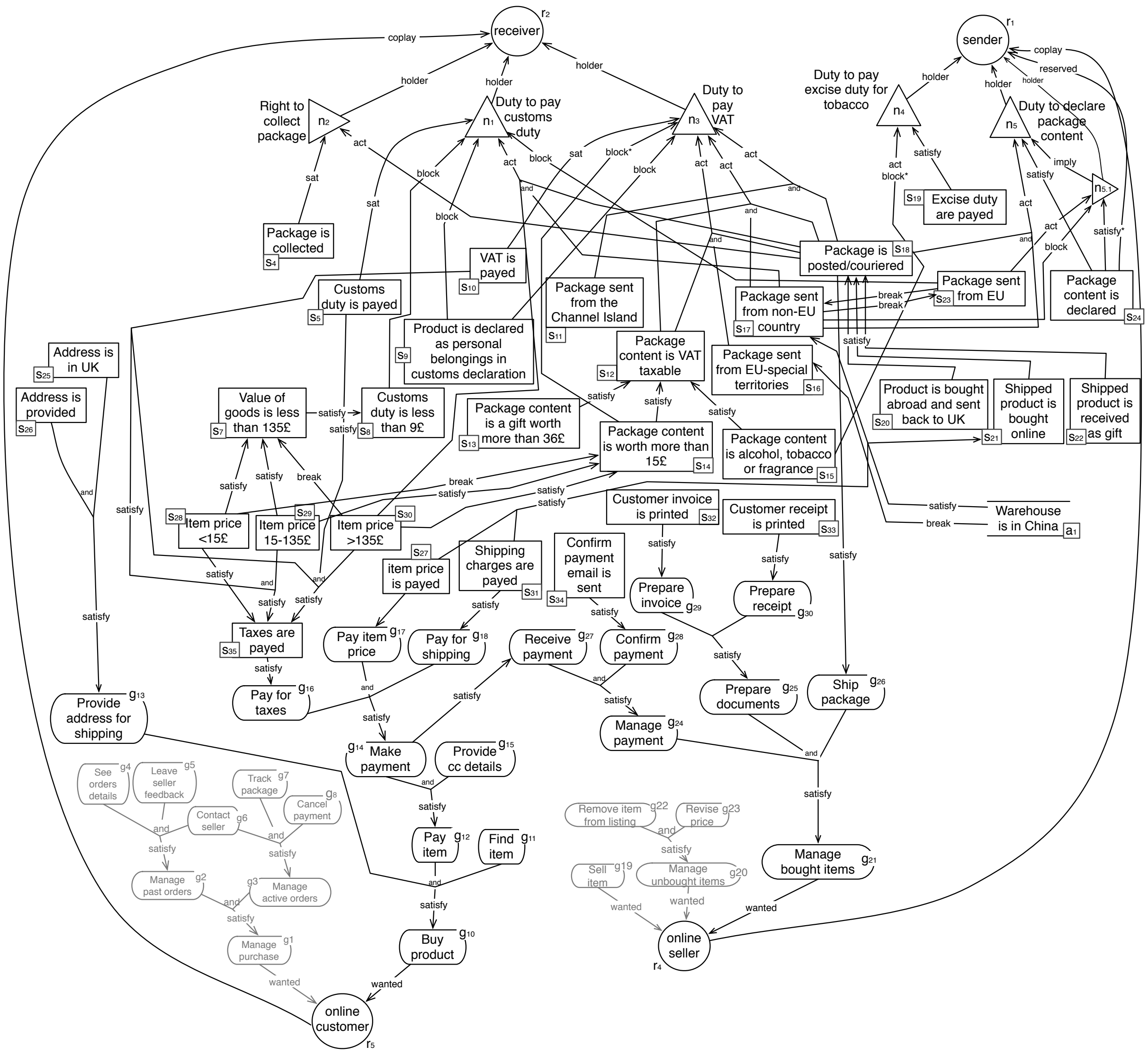


Figure 9.1: The Nòmos 3 model for the example of a software for online sellers to help evaluate taxes and documents for shipping goods to U.K.

9.2 Systematic process: an illustrative example

The objective of this section is to illustrate the systematic process presented in chapter 6 with the use of an illustrative case study. Like in section 7.2 we will use the legal context of the British legislation regulating tax and customs of goods sent from abroad. Moreover, we will use the same example of a software for a British seller selling electronic goods online. The warehouse from where the seller operates is located in China. The goal of the software is to help the seller manage his goods and automatically prepare the necessary paperwork for shipping his goods to U.K. and identify the correct taxes that need to be payed for the goods.

The input to our procedure is a Nòmos 3 model of the requirements, and Nòmos 3 model of the law. For simplicity in the section we will show the two models together in a single diagram: figure 9.1 shows the Nòmos 3 legal and initial goal model for the online seller example with a focus on the features of the software that allows the user to prepare shipping documents and include in the price the right taxes. Throughout the section we will show the complete Nòmos 3 model, however in a real-settings scenario the analyst only uses the result of the legal analysis to operate in the requirements model.

9.2.1 Analysis phase

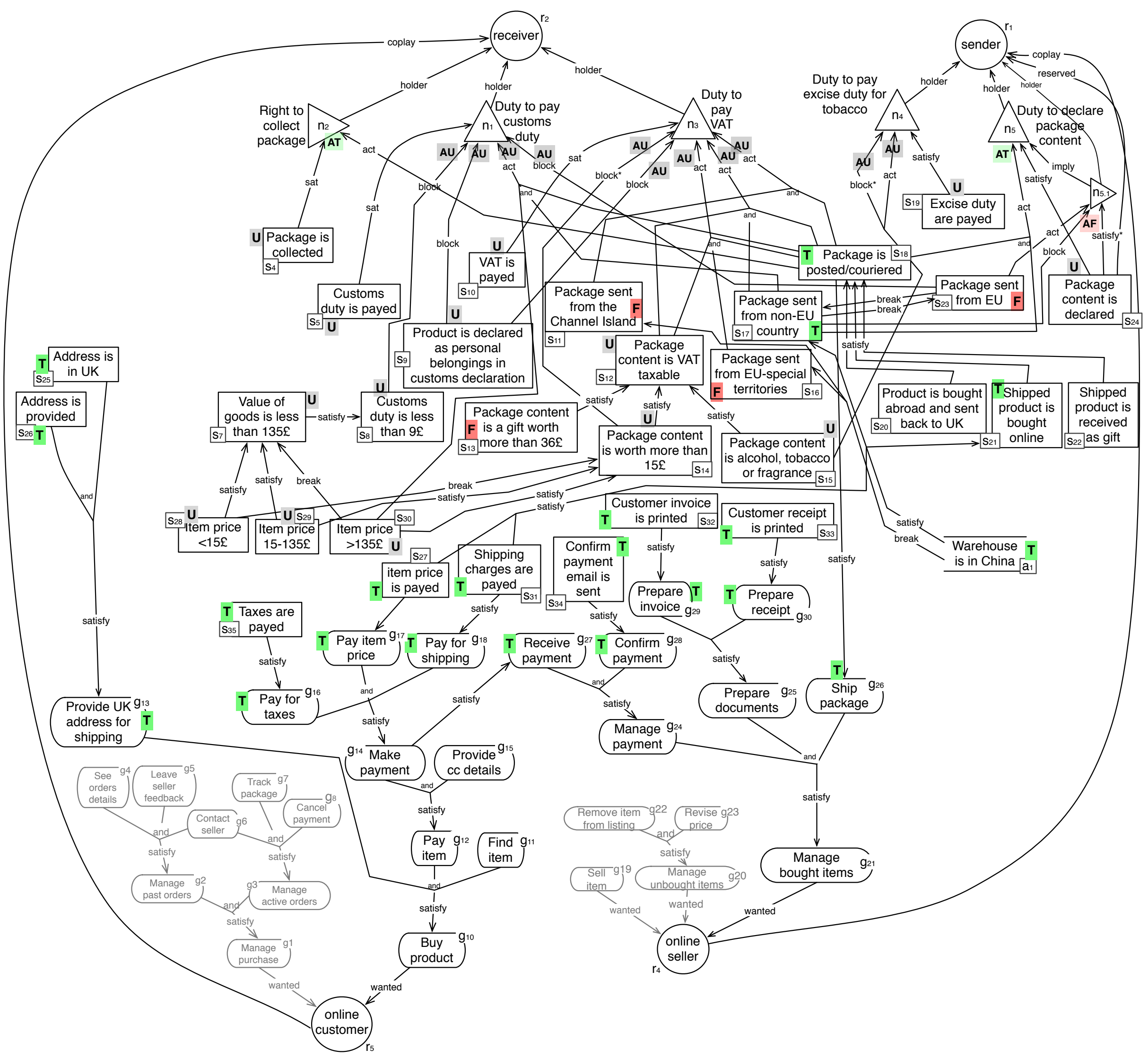
The objective of the analysis phase is to allow the analyst perform an initial evaluation of the requirements model and assess the normative context applying. The requirements models in input to the process is first analyzed to evaluate the Norms that apply to it (figure 9.2). The analysis however reveals that the package receiver has the right to collect the package since it has been shipped, however it is inconclusive whether the other duties to pay taxes are applicable (n_1, n_2). Similarly the sender does not know whether the duty to pay excise duty applies (n_4), but it is known that the duty to declare package content is applicable. The analyst decides to analyze the applicability of the previous Norm in order to evaluate how not to pay excise duty. The `applicability-search()` algorithm informs the analyst that the duty to pay excise duty is only when tobacco, alcohol or fragrance are shipped. Since the seller sells only electronic goods, the analyst can avoid the duty to be applicable by adding a domain assumption that the business does not sell tobacco, alcohol or fragrance. This domain assumption would make $s_{15} = \text{SF}$ ('Package content is alcohol tobacco or fragrance'), which in turns make the duty n_4 not applicable $(s_{15} \xrightarrow{\text{block}*} n_4)$. The analysis of roles does not reveal much more about the Norms applicable to the socials and legal roles, as most of them remain either inconclusive or violated. The analyst decides to go ahead with the next step and use the Revision Phase to be guided on how to revise the inconclusive or violated Norms.

9.2.2 Compliance evaluation phase

The objective of this phase is to evaluate the compliance of the requirements model and, if needed, revise the compliance target. The three activities are detailed in section 6.2.2 and figure 6.3, and are the following:

1. **Define compliance target.** During this activity the analyst has the possibility to specify the desired compliance value for some Norms, or keep the default value where all Norms have value *com*/*tol*. In the current scenario we consider that the analyst maintains as target values the default.

2. **Compliance evaluation.** The second activity is the execution of the compliance algorithm `full-compl-analysis()` to evaluate the compliance value of the Norms in the



receiver
sender
online customer
online seller
Right to collect package
Duty to pay customs duty
Duty to pay VAT
Duty to pay excise duty for tobacco
Duty to declare package content
Package is collected
Customs duty is payed
VAT is payed
Product is declared as personal belongings in customs declaration
Package sent from the Channel Island
Package content is VAT taxable
Package sent from EU-special territories
Package sent from non-EU country
Package is posted/couriered
Package sent from EU
Excise duty are payed
Package content is declared
Address is in UK
Address is provided
Value of goods is less than 135£
Customs duty is less than 9£
Package content is a gift worth more than 36£
Package content is worth more than 15£
Package content is alcohol, tobacco or fragrance
Product is bought abroad and sent back to UK
Shipped product is bought online
Shipped product is received as gift
Item price <15£
Item price 15-135£
Item price >135£
item price is payed
Shipping charges are payed
Confirm payment email is sent
Customer invoice is printed
Customer receipt is printed
Warehouse is in China
Taxes are payed
Pay item price
Pay for shipping
Receive payment
Confirm payment
Prepare invoice
Prepare receipt
Prepare documents
Ship package
Manage payment
Provide UK address for shipping
Pay for taxes
Make payment
Provide cc details
Pay item
Find item
Buy product
Manage bought items
See orders details
Leave seller feedback
Track package
Cancel payment
Contact seller
Manage past orders
Manage active orders
Manage purchase
Remove item from listing
Revise price
Sell item
Manage unbought items

Figure 9.2: The applicability analysis Nòmos 3 model for the example of a software for online sellers to help evaluate taxes and documents for shipping goods to U.K.

model, given the set of requirements. In figure 9.3 we show the value that are propagated to the Norms and their compliance value.

3. **Compare compliance values.** In the last activity the compliance value obtained in the model are compared with the desired one. In table 9.2 we have compared the obtained and desired values.

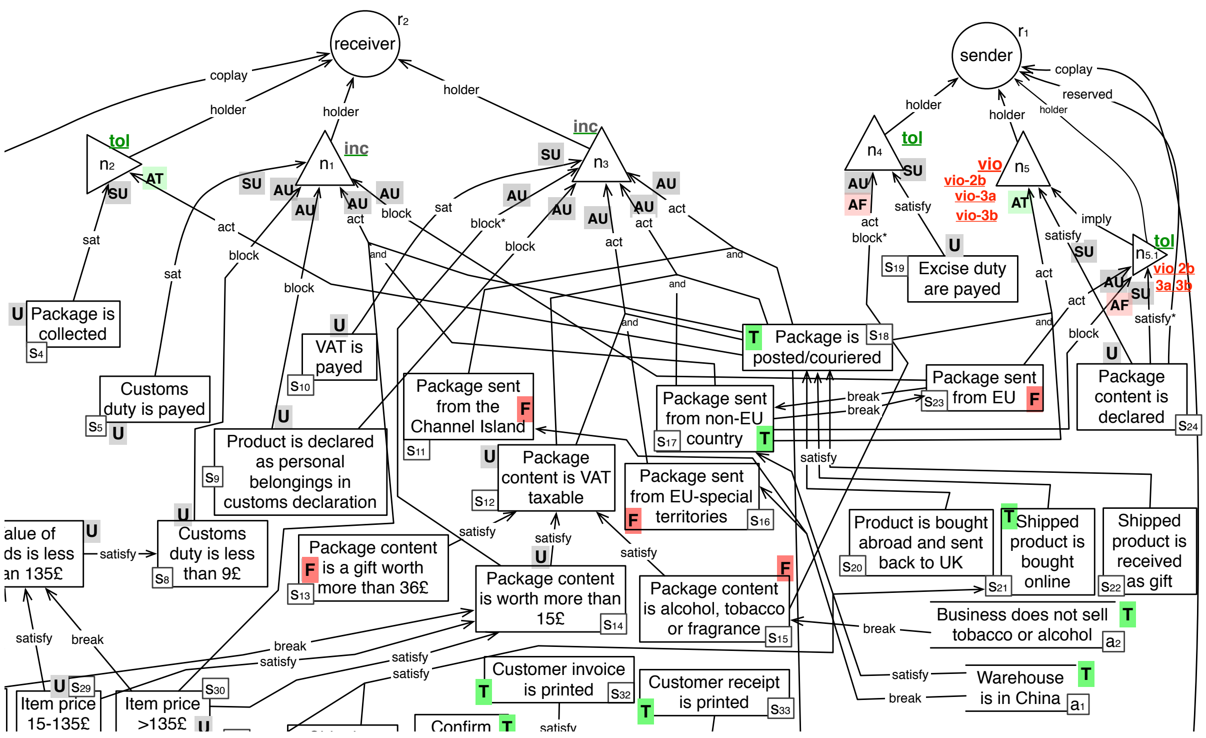

receiver
r2
sender
r1
coplay
holder
reserved
tol
inc
n2
n1
n3
n4
n5
n5.1
AT
SU
AU
AF
vio
vio-2b
vio-3a
vio-3b
vio 2b 3a 3b
sat
act
block
block*
satisfy
imply
and
break
Package is collected
Customs duty is payed
VAT is payed
Product is declared as personal belongings in customs declaration
Package sent from the Channel Island
Package content is VAT taxable
Package sent from non-EU country
Package sent from EU-special territories
Package is posted/couriered
Excise duty are payed
Package sent from EU
Package content is declared
Customs duty is less than 9£
Package content is a gift worth more than 36£
Package content is worth more than 15£
Package content is alcohol, tobacco or fragrance
Product is bought abroad and sent back to UK
Shipped product is bought online
Shipped product is received as gift
Business does not sell tobacco or alcohol
Warehouse is in China
Customer invoice is printed
Customer receipt is printed
Item price 15-135£
Item price >135£
Confirm
T
F
U

Figure 9.3: The compliance evaluation phase of the example of a software for online sellers to help evaluate taxes and documents for shipping goods to U.K.

9.2.3 Revision phase

Lastly, the objective of the revision phase is to help the analyst find a strategy to revise the requirements model. The phase consist of an iteration of two steps that are applied to every Norm not meeting its desired compliance standard in the previous phase. First a revision strategy is selected according to the type of discrepancy between compliance values, and then the analyst is guided through a series of options that can be implemented in the model.

In the following we consider first the Norms that are violated, and then those inconclusive.

1. $\boldsymbol{n_5}$ The duty n_5 to declare the package content has compliance value $vio-2b, vio-3a, vio-3b$. Since the Norm's violations are caused by the reserve schema, the activity of choosing the right revision strategy guides the analyst in fixing the schema by using the mitigation rule suggested by the algorithm `full-compl-search()`. First the analyst can revise the model and adds the missing relationship in the requirements model: $s_{24} \xrightarrow{\text{reserved}} r_4$. With this correction the first violation $vio-2b$ of the schema is resolved (figure 9.4(a)). Subsequently the analyst is guided through another iteration of the revision phase to solve the other violations for the Norm. The two violations $vio-3a$ and $vio-3b$ are also caused by a violation of the reserve schema, and more precisely by the lack of operationalization of the Situation s_{24} in the requirements model (figure 9.4(b)). Similar to the previous case,

Norm	Obtained compl. value	Desired compl. value	
n_1	inc	*com*/*tol*	✗no
n_2	tol	*com*/*tol*	✓ok
n_3	inc	*com*/*tol*	✗no
n_4	tol	*com*/*tol*	✓ok
n_5	vio, vio-2b,vio-3a, vio-3b	*com*/*tol*	✗no
$n_{5.1}$	tol, vio-2b,vio-3a, vio-3b	*com*/*tol*	✗no

Table 9.2: Comparison of the compliance values for the Norms in the the example of a software for online sellers.

the `full-compl-search()` algorithm suggests the mitigation rules of having that Situation satisfy a Goal in the model wanted by the online seller. Since the current goal model only has two Goals related to the preparation of documentation (g_{29} to prepare invoice, and g_{30} to prepare receipt), the analyst decides to add a Goal for the online seller r_4 to 'Prepare declaration of package content' (figure 9.4(c)). With this correction the following intermediate compliance evaluation reveals that all the violations have been solved. By bringing about the Situation, the Norm is also satisfied and therefore compliant.

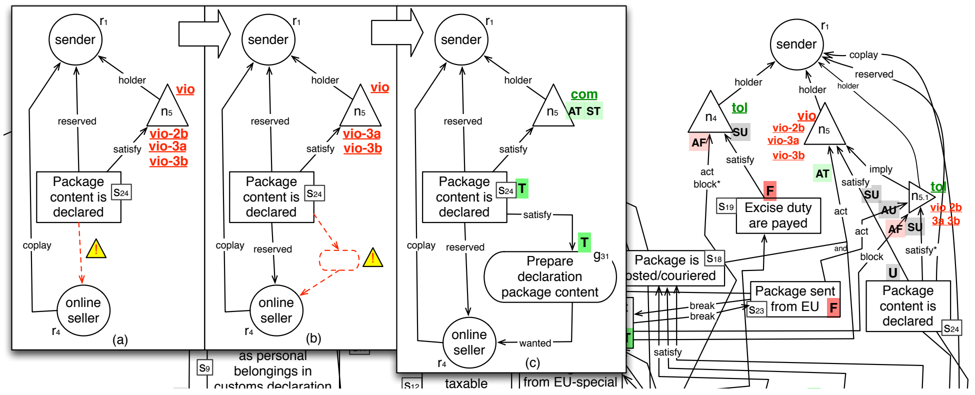


Figure 9.4: The first revision of the model resolves the violation on Norm n_5 that was triggered by the reserve schema.

2. $\boldsymbol{n_{5.1}}$ This right not to declare the package content was initially also violating the reserve schema for the same reasons of Norm n_5. After the corrections implemented the Norm is however evaluated as tolerated. Intuitively, with the revision of the model implemented in the previous step, the analyst had to comply with the duty to declare the package content as the package was shipped from outside the EU.

3. $\boldsymbol{n_3}$ The duty n_3 to pay VAT has inconclusive compliance value (see figure 9.5(a)). A Norm's inconclusiveness is caused by the lack of evidence supporting its applicability or

not applicability, so the process requires the analyst to use the `compliance-search()` or `applicability-search()` algorithms and revise the model in oder to provide some evidence for the Situations responsible for the Norm's applicability and compliance. The study of the applicability for Norm n_3 informs the analyst of the different Situations influencing the Norm's applicability: the Norm is not applicable if the product is declared as personal belongings or its content is worth less than £15, otherwise the duty to pay VAT is applicable. Since the good sold is not personal belongings, the analyst therefore chooses the strategy of 'surrendering to the Norm' and revises the requirements model in such a way as to make sure that if the item costs more than £15, VAT taxes are payed. In figure 9.5(b) the requirements model has been modified by including the Goal $g_{16.3}$ to "*Pay VAT for item >£15*", so when this Goal is achieved VAT is payed (s_{10} = ST) for an item with price higher than £15, which makes the package content VAT taxable (s_{12} = ST), which in turns makes the Norms applicable and satisfied, i.e., compliant.

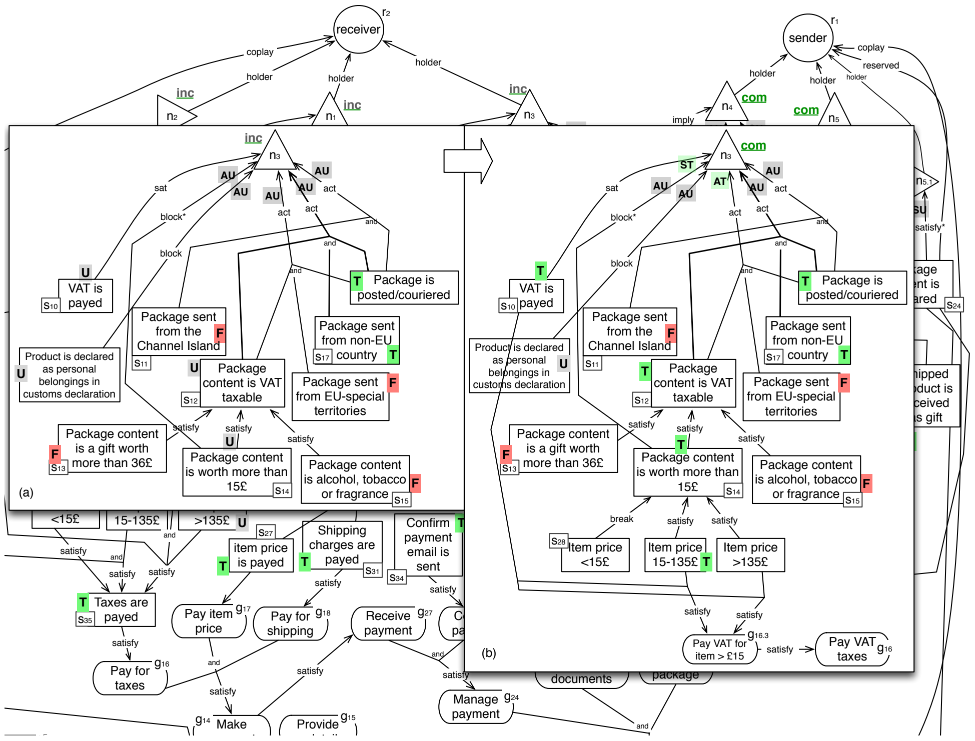

receiver
sender
coplay
holder
reserved
inc
com
n3
sat
block*
block
act
and
AU
ST
AT
VAT is payed
Package sent from the Channel Island
Product is declared as personal belongings in customs declaration
Package content is VAT taxable
Package sent from non-EU country
Package is posted/couriered
Package sent from EU-special territories
Package content is a gift worth more than 36£
Package content is worth more than 15£
Package content is alcohol, tobacco or fragrance
Item price <15£
Item price 15-135£
Item price >135£
Pay VAT for item > £15
Pay VAT taxes
Taxes are payed
Pay for taxes
Pay item price
Pay for shipping
Receive payment
Manage payment
break
satisfy
(a)
(b)

Figure 9.5: The second revision of the model resolves the violation on Norm n_3.

4. $\boldsymbol{n_1}$ The duty n_1 to pay custom duty is also initially evaluated as inconclusive. Similarly to the previous case, the process guides the analyst in understanding the Norm's applicability and compliance with the two algorithms `compliance-search()` and `applicability-search()`. The study of the Norm's applicability informs the analyst of the different possibilities. If customs duty is less than £9 — i.e., when the value of the good is less than £135 — custom duty don't need to be payed. Otherwise — when the

value of the good is higher than £135 — custom duties need to be payed. The analyst decides to revise the model accordingly and ‘surrender to the Norm’. Since the variability of paying the taxes depends on the item’s price, the model is revised in order to correctly reflect the tax(es) to be payed for each category of prices. The analyst decides that the Goal g_{16} ‘Pay taxes’ can be satisfied in three possible ways.

(a) The first (figure 9.6(a)) way would be to $g_{16.1}$ ‘Pay no taxes for item less than £15’.

(b) The second (figure 9.6(b)) way would be to $g_{16.2}$ ‘Pay VAT taxes for items worth more than £15 but less than £135’.

(c) The third and last way (figure 9.6(c)) would be to $g_{16.3}$ ‘Pay VAT taxes and custom duties for items worth more than £135’.

In all three cases, the Situations satisfying the Goals make the duty n_1 either compliant or tolerated.

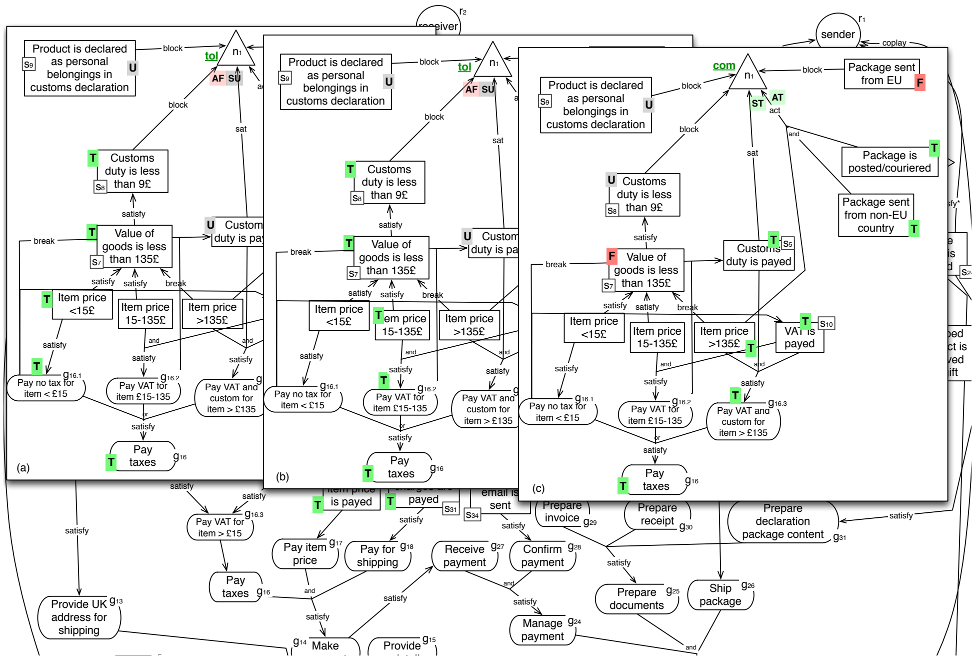

(a)
(b)
(c)
Product is declared as personal belongings in customs declaration
Customs duty is less than 9£
Value of goods is less than 135£
Item price <15£
Item price 15-135£
Item price >135£
Pay no tax for item < £15
Pay VAT for item £15-135
Pay VAT and custom for item > £135
Pay taxes
Package sent from EU
Package is posted/couriered
Package sent from non-EU country
Customs duty is payed
VAT is payed
tol
com
n1
block
satisfy
break
and
sat
act
AF SU
ST AT

Figure 9.6: The second revision of the model resolves the violation on Norm n_1.

5. $\boldsymbol{n_2}$ The right n_2 was initially evaluated as tolerated and needed no further revision.

Once the revision phase is finished, the final model is analyzed again (figure 9.7). Both the analysis of applicable Norms and the analysis of roles are satisfactory since no violations or inconclusive Norms are identified. After the analysis the last compliance evaluation phase is performed, showing that the defined compliance target has been met: the obtained compliance value for all Norms is either complied with or tolerated. The compliance evaluation therefore terminates the process and returns the final requirements model the is represented in figure 9.8

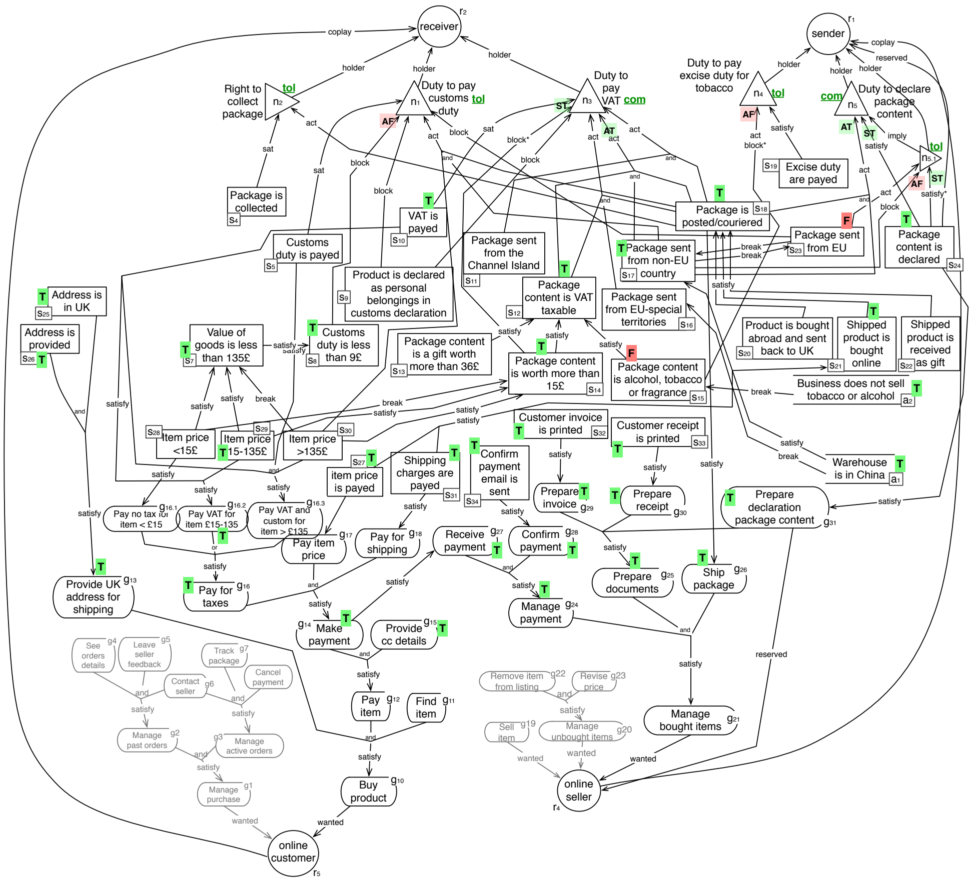


Figure 9.7: The Nòmos 3 legal model and requirements model with all corrections implemented.

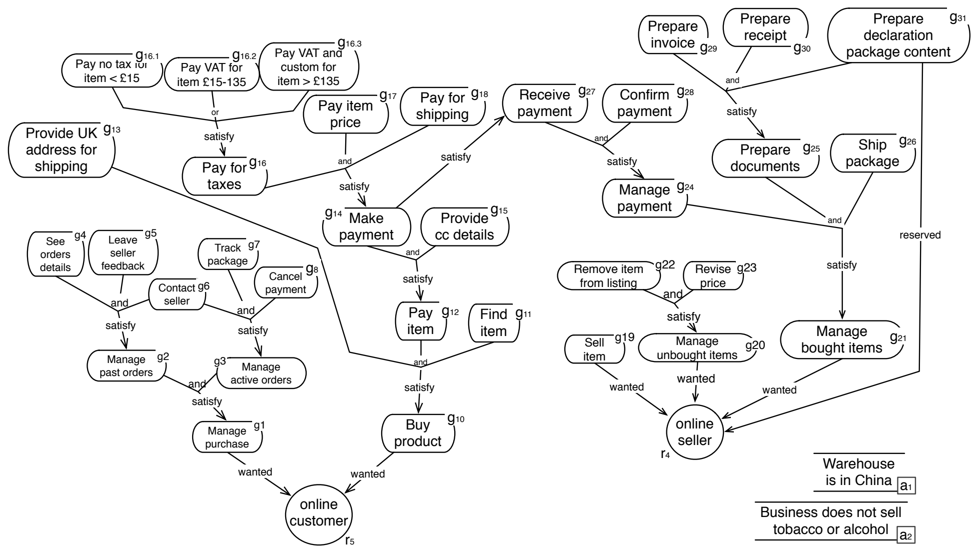


Figure 9.8: The final Nòmos 3 requirements model compliant with all applicable laws.

9.3 Conclusions

The objective of this chapter was to present our findings in the evaluation of the systematic process to help the analyst evaluate the compliance of a set of requirements with a given law (chapter 6).

The preliminary evaluation presented in the first part of the chapter, suggested that the initial proposal for the process was adequate and the logical steps — analysis, evaluation, revision — were appropriate. This earlier evaluation highlighted possible improvements that have been implemented the proposed systematic process evaluated with the illustrative example. First, it was found useful to help and guide the analyst choose a tactic for analyzing and revising the model. In our process in fact, we exploit the Nòmos 3 reasoning algorithms to help the analyst evaluate applicable Norms or alternative ways for complying with a given Norm. Secondly, the initial evaluation exposed how the delicate relationship between Legal and Social Role is more appropriately evaluated and established in a context where the legal expert are involved. For this reason the proposed systematic process involves the legal expert when a revision of the requirements calls for new Social Roles or makes a Social Role coplay a new Legal Role. We assume that the evaluation of the *coplay* relationships in the initial requirements model has also been performed with the help of a legal expert. Moreover, while the analyst benefits from the reasoning algorithms evaluating the legal model, he/she does not directly see the legal model but only uses the result of the analysis to operate in the requirements model.

In future work we are planning to perform a new case study and evaluate the effectiveness of the current revision strategies and identify further ways to help the analyst revise the requirements model. Furthermore we will investigate the use of text mining techniques for automatically identify and link the satisfaction of situations and goals in the model.

Chapter 10

Conclusions

This thesis presented the Nòmos 3 modeling language and a systematic process to help the analyst make a set of requirements comply with applicable laws. First we have presented the Nòmos 3 modeling language for modeling and reasoning about legal variability and compliance of a goal model (chapter 3). Secondly we have presented the reasoning algorithms that exploit the semantics of our language to reason about compliance of norms (chapters 4 and 5). Lastly, we proposed a systematic process that uses the reasoning algorithms to help the analyst revise an initial requirements model to comply with some norms (chapter 6). In the last three chapters we have presented an evaluation of these three contributions.

In this chapter we conclude the thesis and outline the current open issues and ongoing works, and lists some limitations and ideas for our future work.

10.1 Contributions to the state-of-the-art

The contributions of this thesis can be grasped from the research questions proposed in chapter 1 (see also section 1.3.1 on Contributions):

RQ1: How can we represent both law and requirements?

RQ2: How to reason about compliance of a set of requirements?

RQ3: How can we facilitate the use of the Nòmos 3 framework?

RQ4: How well does the proposed framework performs when applied to realistic settings?

As mentioned above this thesis addresses these research questions by proposing the Nòmos 3 modeling language and a systematic process to help the analyst make a set of requirements comply with applicable laws. In the following we detail the specific contributions of this thesis.

First we have proposed a modeling language for representing law and requirements. The Nòmos 3 modeling language supports through a minimal set of concepts and relations, the most important features needed to model the law and requirements, and reason about them. The three basic concepts in Nòmos 3 are Situations, Roles, and Norms. The semantics of the relationships among them is used to characterize how bringing about a set of Goals — represented as Situations desired by Roles in the requirements model — makes Norms applicable, satisfied, complied-with or violated. This contribution addresses the first research question about representing both law and requirements.

The semantics of the Nòmos 3 language defines the behaviour of the relationships in the model and defines how label are propagated in the model. To this end, we have proposed a set of algorithms for reasoning about the variability of law and requirements. Reasoning is performed by means of forward and backward algorithms that allow the user to explore the legal and requirements model and answer important questions like: given a set of Situations in the legal model, which Norms are applicable? complied-with or violated? given a set of Goals and Situations in the requirements model, which Norms are applicable or violated? Which Situations should be brought about in the requirements model to make all applicable Norms complied with? Similarly, we have proposed a set of algorithms for reasoning about how the Roles in both domains relate to the compliance of Norms. Social and Legal Roles — i.e., Roles in the requirements and Roles in the law — are linked to Goals and Norms, and the relationship characterizing the link between the two Roles allows the user to investigate the Norms that the Roles in the requirements model have to respect. Lastly, we expanded the algorithms for reasoning about violations of Norms and overall compliance of a set of requirements, by considering the impact of having the correct (or wrong) Role bringing about some Situations that the law specifically assign to specific Roles. To this end we proposed the reserve schema that allows to detect a Norm's violation when a Situations is brought about by a Role who is not entitled to do so. These contributions address the second questions.

We have proposed a systematic process to help the analyst evaluate the compliance of a set of requirements with a given law. The aim of the process was to guide the analyst through the analysis and revision of a given Nòmos 3 model of requirements using the reasoning algorithms proposed before. The process is composed of three phases. Fist the analyst is guided through an analysis of applicable Norms, then the compliance evaluation phase evaluates the compliance status of the norms that are applicable to the input requirements model. The last phase is aimed at helping the analyst find and apply a revision strategy for amending the model and make it compliant with the piece law. This contribution addresses research question RQ3.

Lastly we have performed an evaluation of the proposal. The language has been evaluated with an analysis of the concepts needed by the language for modeling law, and with an illustrative case study. The reasoning also has been evaluated with an illustrative case study, and we performed some scalability experiments with large artificial models of law reporting on the scalability of our approach with real-size laws. The language and reasoning algorithms have been implemented in a command-line tool for the analysis of Nòmos 3 models. The process has also been evaluated with an illustrative case study and a pilot case study. These works contribute to the last research question RQ4.

10.2 Ongoing work and open issues

In this section we outline some ongoing work which addresses some open issues in the area. First we present a formulation of the compliance problem that takes into account alternatives ways for complying and possible preferences among them [Ingolfo et al., 2013]. Then we will present two preliminary works regarding the application of Nòmos 2 for the evaluation of regulatory compliance in adaptive systems and in business processes [Ingolfo and Souza, 2013; Ghanavati et al., 2014b].

10.2.1 Preferred compliance problem

In this section we report on an initial formulation of the compliance problem that takes into account stakeholder preferences for complying [Ingolfo et al., 2013]. Indeed often there are multiple ways to comply with a given law because of variability elements contained in legal texts, such as conditions, exceptions, derogations, alternatives, cross-references, etc. This variability implies that there isn't a single compliance solution to the compliance problem, but rather a space of compliance alternatives. While alternatives in law are equal to the legislator, they may not be equally desirable to stakeholders: some alternatives may fit better existing requirements, while others may cost less to comply with. For example, the Italian law on privacy mandates that systems managing patients record and files, maintain separated medical and administrative data. From a technical point of view this can be achieved in many ways: using two different physical servers for the different data, using two different encrypted databases but in the same physical server, or — regardless from the physical server — implement some cryptographic mechanism so the encrypted medical data are not usable by the process of the administration of the hospital. All these solution may be considered equally desirable from a legal perspective, but to the stakeholder they are not.

In other words, if a software system has to comply with a given law, *how* it complies also defines how well stakeholder requirements are met. So the problem of ensuring regulatory compliance of requirements includes a search for the best way to comply. In our ongoing work [Ingolfo et al., 2013] we have defined the Preferred Compliance Problem (PCP) as the problem of identifying alternative ways to comply with applicable Norms, and comparing these alternatives on the basis of stakeholder preferences.

The "traditional" compliance problem in Nòmos 3 can be formulated as follows: given a set of potentially satisfiable Situations, find a satisfaction value for a subset of Situations such that they satisfy all applicable norms. The limitation of this definition is that, when there are alternative sets of Situations which satisfy the said conditions, this formulation does not compare these alternatives. This lead us to the definition of the concept of "candidate compliance solution", which would be one one possible assignment to the Situations satisfying the 'traditional' compliance problem. To capture the information that some candidate compliance solutions are more desirable than others, we add to Nòmos 2 a set of binary reflexive, antisymmetric and transitive relations $\leq_C \in \mathcal{S} \times \mathcal{S}$, each $\leq_C$ defining a partial order on Situations. Informally, we call these relations preference relations, and we read $\phi \leq_C \psi$ as "ψ is at least as desirable as ϕ according to criterion C". We let $\phi =_C \psi$ abbreviate "$\phi \leq_C \psi$ and $\psi \leq_C \phi$", so that $\phi <_C \psi$ abbreviates "$\phi \leq_C \psi$ and not $\phi =_C \psi$", and informally reads "ψ is strictly more desirable than ϕ according to criterion C". Each criterion C defines a partial order over Situations. Note that adding preference relations to Nòmos 2 does not influence the satisfaction values, and other features of that language.

Preference relations allow us to record relative desirability of stakeholders between Situations, according to different criteria for comparison. Let $\mathcal{C}$ denote the set of all criteria. We can further add relations between criteria, to help comparisons. We can define a hierarchy of domain-specific criteria for comparison, such as, for example: Criterion Cost is an aggregate of criteria Production cost, Infrastructure cost, Transportation cost, etc. Such a structuring can help define aggregation functions and/or procedures to automatically rank alternative sets of Situations.

The presence of two or more candidate compliance solutions, to a given compliance problem, and the availability of preferences leads to the Preferred Compliance Problem, PCP here after. In contrast to compliance problem, where the aim is to identify a (or at least one) candidate compliance solution, the PCP requires that preference be used to select one candidate compliance solution, as the compliance solution to the PCP.

The PCP can be stated as follows: Given a set of potentially satisfiable Situations $\mathcal{S}$, find a set of Situations $X \in \wp(\mathcal{S})$, such that 1) X make some Norms N applicable and satisfied, and 2) there is no set of Situations X' such that the candidate compliance solution (X', N) ranks higher than candidate compliance solution (X, N), according to a given ranking function r, which returns a total order over all Candidate Compliance Solutions.

10.2.2 Regulatory compliance in adaptive systems

In this section we report on very early work, on the application of adaptive software system design techniques to the problem of designing compliant software[Ingolfo and Souza, 2013].

Uncertainty from the environment and variability of solutions are amongst the topics of study in the research area of adaptive software system design Cheng et al. [2009]. Adaptive systems treat this uncertainty in a dynamic way, changing the system's behavior to a different variant when needed. Likewise, the law needs to be handled dynamically in order to accommodate differences in legislation and changes in regulations. For instance, in an autonomous car, drivers could indicate the maximum amount of dollars they are willing to spend for speeding tickets, and then driving across different states the car settings adapt according to applicable limits and fines.

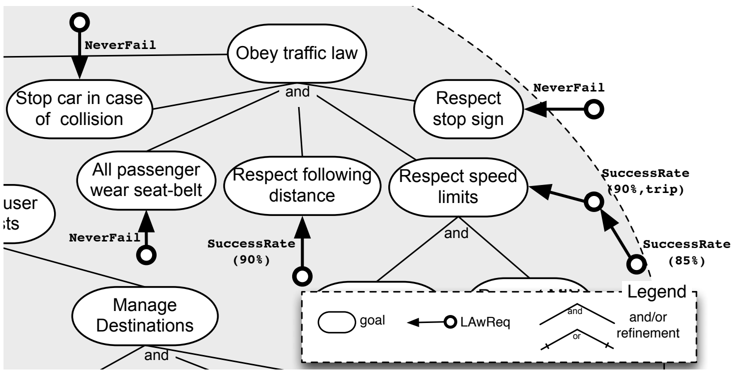


Figure 10.1: LAwReq examples elicited based on the autonomous car model [?].

In the following we report our preliminary work on the application of adaptive software system design techniques to the problem of designing software in compliance with some laws. Our approach is founded on concepts adopted from Requirements Engineering (RE) and uses the *Zanshin* framework [Souza et al., 2011, 2012] for the design of adaptive systems. *Zanshin* is based on the idea that adaptivity is implemented by a monitor-adapt feedback loop that reads from the system's requirements model what should be monitored and what to do in case monitoring indicates failures. These are represented in the model by *Awareness Requirements* (*AwReqs*) and *Evolution Requirements* (*EvoReqs*), respectively. On the monitoring side, *AwReqs* represent constraints on the states that other requirements can assume during their execution at runtime.

For example in the scenario of the design of an autonomous car, we can consider a piece of software (called 'the vehicle operator') that manages the car in all its basic features (drive in the traffic, obey traffic rules, etc.) and takes care of the user request. In this settings the vehicle operator has to obey the traffic laws and, consequently, its specifications also include legal requirements such as the goal *Obey traffic law* and its refinements. The feedback loop mechanism applies to both types of requirements as we need to be sure that also the legal prescriptions that apply to our system are respected. In this spirit, we categorize these *AwReqs* as *Legal Awareness Requirements* (*LAwReqs*): the class of legal requirements that lead to feedback loop functionalities. In other words, *LAwReqs* will talk about the states that *legal requirements* — software requirements coming from the law — can assume at runtime.

Figure 10.1 shows a few examples of *LAwReqs*, represented using the same syntax as regular *AwReqs* (cf. Souza et al. [2011]). Regulations that are identified as being more critical — such as stopping in case of collision, wearing seat-belts and respecting stop signs — are associated with "never fail" constraints, whereas less critical parts of the law have their deniability (failure) tolerated up to certain points: the car should respect both the following distance and the speed limits 90% of the time. For more details on how *Zanshin* operationalizes monitoring, refer to Souza et al. [2011].

On the adaptation side, *EvoReqs* prescribe what to do in case an *AwReq* fails, in order to adapt the system. As before, this component of the feedback loop could be applied to *LAwReqs* as well, specifying counter-measures whenever the system has reached a level of noncompliance with the law that should not be tolerated. Adaptation strategies can consist of precise actions, including changes in the model itself. For example, if a passenger is not wearing her seat-belt, the autonomous car first issues a warning, then, if the problem persists, it can decrease its speed and ultimately park itself, resuming the trip only when the situation is resolved.

Another strategy for adaptation is to look for a *system parameter* that can be reconfigured the same way a control system (e.g., a thermostat) tunes its variables (the heating/cooling power) to keep its output (the room temperature) as close as possible to a desired value. The speed of the car, the driving style (e.g., conservative, aggressive), preferable routes (e.g., highway, in the city), etc. are examples of possible parameters for the autonomous car.

In the above, we have seen legal requirements as targets for monitoring and adaptation. However, the law can also be used as a source in the elicitation of these requirements for adaptation. In the second part of this ongoing work we have sketched an approach for the design of adaptation requirements based on legal documents.

High-level methodology for adapting to the law. An important aspect of software adaptation comes from the identification of information representing the relationship among the parameters of the system and indicators that it is operating properly (in *Zanshin*, for example, *AwReqs* represent such indicators). The inclusion of these elements in the requirements model provides an essential link between the possible system configurations and its measured output, supporting the design of adaptation features. The main claim of this preliminary work is that legal texts can be useful sources for identifying new indicators of requirements convergence and parameters that can be tuned at runtime to help maintain such indicators close to desired reference values. To this end, we have proposed a preliminary methodology that takes as input a goal model of the requirements of the system and a piece of law and returns the requirements model expanded with parameters and indicators coming from the legal text. The main steps of the methodology are the following.

1. The first step of the methodology is to *identify the object regulated* in the analyzed piece of law at hand. We call *object* the element that the law is regulating and for which rules and limitations are in place, e.g., the driver's blood alcohol content (BAC), the number of passengers, the speed of the car, the distance of the car from an object, the way a car can/cannot be operated, etc.

2. As second step we need to evaluate if the *object is measurable and quantifiable* (in general), and if it can be measured by the system. Once again, specific automatic techniques could be of help and support this task with suggestion to the analysts. In the example above, the amount of hours can be measured by a piece of software and no special equipment would be needed in a car in order to quantify the number of driving hours.

3. As third and last step we need to evaluate, for each object analyzed in the previous step, *if and which adaptation-related elements to include* in the requirements model.
   If the object is measurable by the system and is relevant enough to be used as an indicator, a *LAwReq* can be added to the model. For example, in a car with a device to measure blood alcohol level, the information can be used by the system to perform different types of adaptation.
   If an object cannot be directly measured by the system or a solution to do it is not feasible then a proxy solution could be envisaged by the analyst in order to evaluate the legal object. For example, if the car does not have a device to measure blood alcohol level, then the software could ask the user for the number of drinks.
   Moreover, System parameters can become system parameters if they have an effect on the success rate of an *AwReq*. For instance, other articles from the California Vehicle Code mention the car's current speed. Clearly, the value of this particular element affects the satisfiability of goal *Respect speed limits* and, thus, the *LAwReq* associated to it.
   Lastly, the analysis of legal text can also lead to the precise adaptation strategy to be used in case a *LAwReq* is not satisfied, leading to the inclusion of *EvoReqs* in the requirements model. *LAwReqs* can also be set up in order to prevent the system from breaking the law instead of just remedying the situation. For instance, a different domain assumption considering not 10, but 8 or 9 driving hours could be elicited with an associated *LAwReq* that, when not satisfied, would configure the car with a more cautious setting (e.g., using lower speeds) and periodically notify the driver until it reaches the limit of hours.

10.2.3 Nòmos 2 for compliance of Business Processes

In this section we report on a recent preliminary work on the use of Nòmos 2 (the legal reasoning in Nòmos 3) for the evaluation of compliance of business process Ghanavati et al. [2014b]. The core idea behind this work is that business processes can be very large and articulated, and at execution time they generate a potentially large number of alternative execution paths. A business process consists of a set of activities to be executed in sequence. Gateways, such as decision points and parallelisms, split the activity sequence into different possible flows, potentially executed in parallel by multiple actors (lanes). The presence of parallelism in particular, generates a sort of indeterminism in process execution, which in turn causes multiple execution instances to be possible, depending on the concurrency conditions. Laws are similarly composed of a large set of conditional elements, such as conditions, exceptions and so on, which create an even potentially larger number of admissible paths to comply. The objective of the Nòmos 2 modeling language was indeed to support reasoning over these large legal variability.

To evaluate the compliance of a business process we have proposed an approach for the exploration of business process paths to detect potential violations of the law with Nòmos 2. We defined the concept of a *legal business process path* as a path in the process such that the

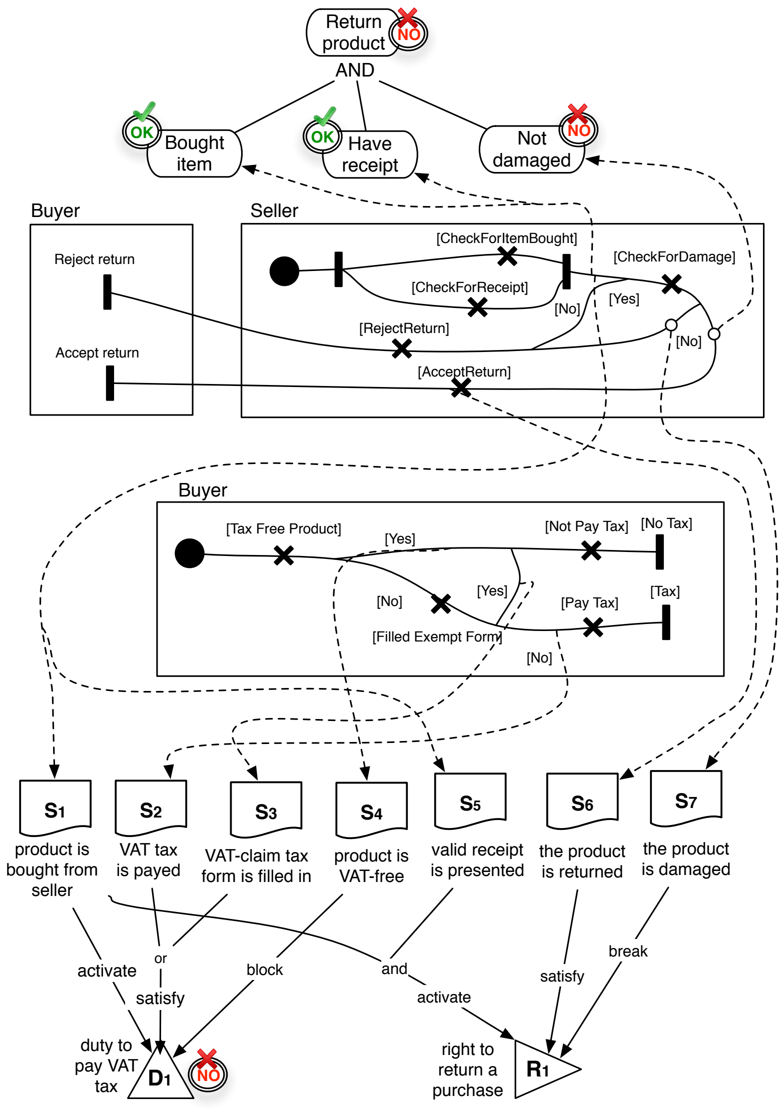


Figure 10.2: Reasoning with a Nòmos 2 legal model in URN [Ghanavati et al., 2014b].

supported requirements are satisfied, while the applicable norms are not violated.

The proposed approach uses URN and Nòmos 2 to enforce modeling business processes that satisfy stakeholder requirements and at the same time comply with applicable laws. In particular, URN offers basics to model requirements with GRL and business processes with UCM, and allows to establish traceability links between them. Our approach lies on the idea that adopting similar traceability links between business processes and the Nòmos 2 model of law, we can perform exhaustive search in the space of traces generated by a given business process.

Figure 10.2 depicts an example of our approach using the scenario of an electronic commerce software where compliance with tax laws is considered. The example represents two actors — the Buyer and the Seller — and two business processes: one process involving both actors, and one only with the Buyer. Since the two processes run in parallel, there is no way to ensure that a certain flow of activities is executed. While we can ensure that when each activity is performed — and its corresponding requirement is satisfied — it is possible that performing a combination of activities results in a violation. To represent a satisfiability conditions between the business process and the law, we use traceability links from the path branch of the business process to the corresponding Situation in Nòmos 2 model. For example, in the business process related to the tax law (the second in the figure), two conditional verifications must happen. First, it is necessary to check if the product is tax free. If it is, Situation s_4 is satisfied. Next, if the product is not tax-free, the second check has to be done to verify whether the buyer filled up the tax exemption form or not. The two paths created from this satisfy s_2 and s_3 in Nòmos 2. In the process of 'returning the product' (the first process in the figure), the seller only accepts the product as returned if three conditions (i.e. having bought the product, having valid receipt, and returning the purchased product with no damage) are satisfied. These three conditions satisfy the Situations, s_5, s_6 and s_7 in Nòmos 2 respectively. When the buyer does not satisfy all three conditions, the main goal 'Return product' will not be satisfied, as shown in figure 10.2.

10.3 Limitations and future directions

In this section we outline some limitations of our work and highlight future directions for our research.

- Despite the fact that we have not had the chance to perform validation in an industrial context, we nonetheless recognize its importance and indeed will work in this direction as part of our future work. Moreover, throughout the development of the framework we have had the opportunity to meet and informally discuss with legal experts our approach, providing us with a basic assessment and opinion of our work. Similarly, it is needed a more rigours evaluation of the accuracy and completeness of the legal reasoning offered by Nòmos 3. To address this limitation we are planning on performing a controlled case study to collect feedback from experts in the legal domain in order to evaluate our reasoning from a legal standpoint. This evaluation would allow us to measure the alignment between the reasoning of legal experts and with that offered by Nòmos 3.

- An important limitation of our work is the generation of Nòmos 3 models of law. Laws and regulations often consists of hundreds of pages of texts, so it is reasonable to assume that the corresponding Nòmos 3 legal models would also be really large and be made of thousands concepts and relationships. Presently our models are built manually and the effort needed to analyze the text and build the model is not negligible. Our future plans include exploring how to exploit existing tools for legal text analysis to support the extraction of legal models from text.

Moreover many of the peculiarities present in the legal texts can not be automatically fixed or implied. For example, during the analysis of the Italian Guidelines on Electronic Health Record (see section 7.1) we identified several problems in the text that needed interpretation: not only the ambiguity of terms, but also the lack of specification of the holder of some Norms, the lack of specificity with respect to what made the Norm applicable, and so on. These limitations are important indications that these models necessarily need the involvement of legal experts in their creations/amendments.

- Our approach concentrates on goal models as representation of the requirements model, and the important link in the Nòmos 3 models concerns the identification of the Situations that needs to be brought about for a given Goal to be considered satisfied. This link between Goals and Situations is taken as assumption in our work however future work will need to investigate techniques for accurately assess the set of Situations that need to hold/not hold, in order to consider a set of Goals satisfied.

- The command-line tool — NRTool— has been developed with the purpose of experimenting with our reasoning and use it during our scalability experiment. Its implementation is however still a prototype: further development is needed in order to accept also as input some graphical Nòmos 3 models and allow the user to draw model and analyze them within the tool.

- The positive results of the scalability study of our reasoning suggests that our proposal would scale to real-sized law, however the use of artificial models limit the scope of our findings. One possibility for our future work would be to investigate the use of seed model that are not artificial but rather larger models from real laws. Through the replication of the this initial model, a more realistic structure of the overall law is created.

Appendix A

NRTool configuration file

The command-line reasoning tool uses a configuration file containing the rules to convert a description of the Nòmos 3 model into a Datalog file. The tool uses the Apache Velocity[1] language to create the Datalog file of the model given in input.

```
% Satisfiability rules for situation $s.getId()
#foreach( $s in $program.getModel().propositions( "situation" ) )
st($s.getId()) :- stx($s.getId()).
su($s.getId()) :- sux($s.getId()), not stx($s.getId()).
sf($s.getId()) :- sfx($s.getId()), not sux($s.getId()), not stx($s.getId()).
#end

% Applicability and satisfiability rules for norm $s.getId()
#foreach( $s in $program.getModel().propositions( "norm" ) )
st($s.getId()) :- stx($s.getId()).
su($s.getId()) :- sux($s.getId()), not stx($s.getId()).
sf($s.getId()) :- sfx($s.getId()), not sux($s.getId()), not stx($s.getId()).
at($s.getId()) :- atx($s.getId()).
au($s.getId()) :- aux($s.getId()), not atx($s.getId()).
af($s.getId()) :- afx($s.getId()), not aux($s.getId()), not atx($s.getId()).
#end

% Compliance rules for right $s.getId()
#foreach( $s in $program.getModel().propositions( "norm" ) )
#if( $s.getModality() == 1 )
com($s.getId()) :- at($s.getId()), st($s.getId()).
tol($s.getId()) :- at($s.getId()), not st($s.getId()).
tol($s.getId()) :- af($s.getId()).
inc($s.getId()) :- au($s.getId()).
#else
% Compliance rules for duty $s.getId()
com($s.getId()) :- at($s.getId()), st($s.getId()).
vio($s.getId()) :- at($s.getId()), not st($s.getId()).
tol($s.getId()) :- af($s.getId()).
inc($s.getId()) :- au($s.getId()).
#end
#end
```

[1] http://velocity.apache.org/.

```
% Fulfillment rules for lrole $s.getId()
#foreach( $s in $program.getModel().propositions( "lrole" ) )
ft($s.getId()) :- ftx($s.getId()).
fu($s.getId()) :- fux($s.getId()), not ftx($s.getId()).
ff($s.getId()) :- ffx($s.getId()), not fux($s.getId()), not ftx($s.getId()).
#end


% Fulfillment rules for srole $s.getId()
#foreach( $s in $program.getModel().propositions( "srole" ) )
ft($s.getId()) :- ftx($s.getId()).
fu($s.getId()) :- fux($s.getId()), not ft($s.getId()).
ff($s.getId()) :- ffx($s.getId()), not fux($s.getId()), not ft($s.getId()).
#end

% Fulfillment rules for lrole $s.getId()
#foreach( $s in $program.getModel().propositions( "lrole" ) )
ft($s.getId()) :- ftx($s.getId()).
fu($s.getId()) :- fux($s.getId()), not ft($s.getId()).
ff($s.getId()) :- ffx($s.getId()), not fux($s.getId()), not ft($s.getId()).
#end


% Satisfiability rules for goal-situation $s.getId()
#foreach( $s in $program.getModel().propositions( "goal" ) )
st($s.getId()) :- stx($s.getId()).
su($s.getId()) :- sux($s.getId()), not stx($s.getId()).
sf($s.getId()) :- sfx($s.getId()), not sux($s.getId()), not stx($s.getId()).
#end


%%%%%% ACTIVATE *********************
% AND ACTIVATE
#foreach( $r in $program.getModel().relations( "activate" ) )
% activate #foreach( $s in $r.getSources() ) $s.getId() #end $r.getTarget().getId()
atx($r.getTarget().getId()) :- #foreach( $s in $r.getSources() )st($s.getId()),#end not st($r.getTarget().getId()s).
aux($r.getTarget().getId()) :- not atx($r.getTarget().getId()), not st($r.getTarget().getId()s).
#end

% OR ACTIVATE
#foreach( $r in $program.getModel().relations( "or-activate" ) )
% or-activate #foreach( $s in $r.getSources() ) $s.getId() #end $r.getTarget().getId()
aux($r.getTarget().getId()):- #foreach($s in $r.getSources()) not st($s.getId()),#end not st($r.getTarget().getId()s).
 #foreach( $s in $r.getSources() )
 atx($r.getTarget().getId()) :- st($s.getId()), not st($r.getTarget().getId()s).
 #end
#end

% ACTIVATE - star - and
#foreach( $r in $program.getModel().relations( "star-activate-and" ) )
% activate-star-and #foreach( $s in $r.getSources() ) $s.getId() #end $r.getTarget().getId()
atx($r.getTarget().getId()) :- #foreach( $s in $r.getSources() )sf($s.getId()),#end not st($r.getTarget().getId()s).
```

```
aux($r.getTarget().getId()) :− not atx($r.getTarget().getId()), not st($r.getTarget().getId()s).
#end

% ACTIVATE − star − or
#foreach( $r in $program.getModel().relations( "star−activate−or" ) )
% activate−star−or #foreach( $s in $r.getSources() ) $s.getId() #end $r.getTarget().getId()
aux($r.getTarget().getId()):− #foreach($s in $r.getSources()) not sf($s.getId()),#end not st($r.getTarget().getId()s).
 #foreach( $s in $r.getSources() )
 atx($r.getTarget().getId()) :− sf($s.getId()), not st($r.getTarget().getId()s).
 #end
#end


%%%%% BLOCK *********************
% AND BLOCK
#foreach( $r in $program.getModel().relations( "block" ) )
% block #foreach( $s in $r.getSources() ) $s.getId() #end $r.getTarget().getId()
afx($r.getTarget().getId()) :− #foreach( $s in $r.getSources() )st($s.getId()),#end not st($r.getTarget().getId()s).
aux($r.getTarget().getId()) :− not afx($r.getTarget().getId()), not st($r.getTarget().getId()s).
#end

% OR BLOCK
#foreach( $r in $program.getModel().relations( "or−block" ) )
% or−block #foreach( $s in $r.getSources() ) $s.getId() #end $r.getTarget().getId()
aux($r.getTarget().getId()):− #foreach( $s in $r.getSources()) not st($s.getId()),#end not st($r.getTarget().getId()s).
 #foreach( $s in $r.getSources())
 afx($r.getTarget().getId()) :− st($s.getId()), not st($r.getTarget().getId()s).
 #end
#end

% STAR BLOCK AND
#foreach( $r in $program.getModel().relations( "star−block−and" ) )
% block #foreach( $s in $r.getSources() ) $s.getId() #end $r.getTarget().getId()
afx($r.getTarget().getId()) :− #foreach( $s in $r.getSources() )sf($s.getId()),#end not st($r.getTarget().getId()s).
aux($r.getTarget().getId()) :− not afx($r.getTarget().getId()), not st($r.getTarget().getId()s).
#end

% STAR BLOCK OR
#foreach( $r in $program.getModel().relations( "star−block−or" ) )
% or−block #foreach( $s in $r.getSources() ) $s.getId() #end $r.getTarget().getId()
aux($r.getTarget().getId()):− #foreach($s in $r.getSources()) not sf($s.getId()),#end not st($r.getTarget().getId()s).
 #foreach( $s in $r.getSources() )
 afx($r.getTarget().getId()) :− sf($s.getId()), not st($r.getTarget().getId()s).
 #end
#end


%%%%% SATISFY *********************

% AND SATISFY
#foreach( $r in $program.getModel().relations( "satisfy" ) )
% satisfy #foreach( $s in $r.getSources() ) $s.getId() #end $r.getTarget().getId()
stx($r.getTarget().getId()) :− #foreach( $s in $r.getSources() ) st($s.getId())#if(($r.sources.indexOf($s)+1) == ($r.
     getSourceCount()) ).
 satisfy ($s.getId(),$r.getTarget().getId()).
#else,#end#end
sux($r.getTarget().getId()) :− not stx($r.getTarget().getId()), not st($r.getTarget().getId()s).
#end

% OR SATISFY
#foreach( $r in $program.getModel().relations( "or−satisfy" ) )
% or−satisfy #foreach( $s in $r.getSources() ) $s.getId() #end $r.getTarget().getId()
sux($r.getTarget().getId()) :− #foreach( $s in $r.getSources() ) not st($s.getId())#if(($r.sources.indexOf($s)+1) ==
      ($r.getSourceCount()) ).
#else,#end#end
 #foreach( $s in $r.getSources() )
 stx($r.getTarget().getId()) :−  st($s.getId()).
 #end
#end

% STAR SATISFY AND
#foreach( $r in $program.getModel().relations( "star−satisfy−and" ) )
% satisfy #foreach( $s in $r.getSources() ) $s.getId() #end $r.getTarget().getId()
stx($r.getTarget().getId()) :− #foreach( $s in $r.getSources() ) sf($s.getId())#if(($r.sources.indexOf($s)+1) == ($r.
      getSourceCount()) ).
#else,#end#end
sux($r.getTarget().getId()) :− not stx($r.getTarget().getId()), not st($r.getTarget().getId()s).
#end

% STAR SATISFY OR
#foreach( $r in $program.getModel().relations( "star−satisfy−or" ) )
% or−satisfy #foreach( $s in $r.getSources() ) $s.getId() #end $r.getTarget().getId()
sux($r.getTarget().getId()) :− #foreach( $s in $r.getSources() ) not sf($s.getId())#if(($r.sources.indexOf($s)+1) ==
      ($r.getSourceCount()) ).
#else,#end#end
 #foreach( $s in $r.getSources() )
 stx($r.getTarget().getId()) :−  sf($s.getId()).
 #end
#end

%%%%%% BREAK **********************

% AND BREAK
#foreach( $r in $program.getModel().relations( "break" ) )
% break #foreach( $s in $r.getSources() ) $s.getId() #end $r.getTarget().getId()
sfx($r.getTarget().getId()) :− #foreach( $s in $r.getSources() )st($s.getId())#if(($r.sources.indexOf($s)+1) == ($r.
      getSourceCount())).
#else,#end#end
sux($r.getTarget().getId()) :− not sfx($r.getTarget().getId()), not st($r.getTarget().getId()s).
#end

% OR BREAK
#foreach( $r in $program.getModel().relations( "or−break" ) )
% or−break #foreach( $s in $r.getSources() ) $s.getId() #end $r.getTarget().getId()
sux($r.getTarget().getId()) :− #foreach( $s in $r.getSources() ) not st($s.getId())#if(($r.sources.indexOf($s)+1) ==
      ($r.getSourceCount())).
#else,#end#end
#foreach($s in $r.getSources())
```

```
sfx($r.getTarget().getId()) :-  st($s.getId()).
#end
#end

% STAR BREAK AND
#foreach( $r in $program.getModel().relations( "star-break-and" ) )
% break #foreach( $s in $r.getSources() ) $s.getId() #end $r.getTarget().getId()
sfx($r.getTarget().getId()) :- #foreach( $s in $r.getSources() )sf($s.getId())#if(($r.sources.indexOf($s)+1) == ($r.
    getSourceCount())).
#else,#end#end
sux($r.getTarget().getId()) :- not sfx($r.getTarget().getId()), not st($r.getTarget().getId()s).
#end

% STAR BREAK OR
#foreach( $r in $program.getModel().relations( "star-break-or" ) )
% or-break #foreach( $s in $r.getSources() ) $s.getId() #end $r.getTarget().getId()
sux($r.getTarget().getId()) :- #foreach( $s in $r.getSources() ) not sf($s.getId())#if(($r.sources.indexOf($s)+1) ==
    ($r.getSourceCount())).
#else,#end#end
#foreach($s in $r.getSources())
sfx($r.getTarget().getId()) :-  sf($s.getId()).
#end
#end

%%%%%% IMPLY *********************
#foreach( $r in $program.getModel().relations( "imply" ) )
%  #foreach( $s in $r.getSources() ) $s.getId() #end imply $r.getTarget().getId()
#foreach( $s in $r.getSources() )
vio($r.getTarget().getId()) :- vio($s.getId()).
inc($r.getTarget().getId()) :- inc($s.getId()), not vio($s.getId()).
tol($r.getTarget().getId()) :- tol($s.getId()), not inc($s.getId()), not vio($s.getId()).
com($r.getTarget().getId()) :- com($s.getId()), not tol($s.getId()), not inc($s.getId()), not vio($s.getId()).
#end #end

%%%%%% DEROGATE *********************
#foreach( $r in $program.getModel().relations( "derogate" ) )
% $r.getTarget().getId()s exists with its  satisfiability  rules
#foreach( $s in $r.getSources() )
st($r.getTarget().getId()s) :- stx($r.getTarget().getId()s).
su($r.getTarget().getId()s) :- sux($r.getTarget().getId()s), not stx($r.getTarget().getId()s).
sf($r.getTarget().getId()s) :- sfx($r.getTarget().getId()s), not sux($r.getTarget().getId()s), not stx($r.getTarget().
    getId()s).
% if source norm is com, then the s-situation (derogate-sit) is true
st($r.getTarget().getId()s) :- com($s.getId()).
su($r.getTarget().getId()s) :- not com($s.getId()).
% the s-situation blocks the norm
% and-block ($r.getTarget().getId(),$r.getTarget().getId()s)
afx($r.getTarget().getId()) :- st($r.getTarget().getId()s).
aux($r.getTarget().getId()) :- not st($r.getTarget().getId()s).
#end #end

%%%%%% ENDORSE *********************
#foreach( $r in $program.getModel().relations( "endorse" ) )
% $r.getTarget().getId()e exists with its  satisfiability  rules
```

```
#foreach( $s in $r.getSources() )
st($r.getTarget().getId()e) :− stx($r.getTarget().getId()e).
su($r.getTarget().getId()e) :− sux($r.getTarget().getId()e), not stx($r.getTarget().getId()e).
sf($r.getTarget().getId()e) :− sfx($r.getTarget().getId()e), not sux($r.getTarget().getId()e), not stx($r.getTarget().
    getId()e).
% if source norm is com, then the e−situation (endorse−sit) is true
st($r.getTarget().getId()e) :− com($s.getId()).
su($r.getTarget().getId()e) :− not com($s.getId()).
% the e−situation activates the norm
% and−activate ($r.getTarget().getId(),$r.getTarget().getId()s)
atx($r.getTarget().getId()) :− st($r.getTarget().getId()e).
aux($r.getTarget().getId()) :− not st($r.getTarget().getId()e).
#end #end


%%%%%% HOLD **********************
#foreach( $r in $program.getModel().relations( "hold" ) )
 #foreach( $s in $r.getSources() )
 stx($r.getTarget().getId()$s.getId()) :− com($s.getId()).
 stx($r.getTarget().getId()$s.getId()) :− tol($s.getId()).
 ffx($r.getTarget().getId()) :− vio($s.getId()).
 fux($r.getTarget().getId()) :− au($s.getId()).
 holder($s.getId(),$r.getTarget().getId()).
 #end
ftx($r.getTarget().getId()) :− #foreach( $s in $r.getSources() ) stx($r.getTarget().getId()$s.getId())#if(($r.sources.
    indexOf($s)+1) == ($r.getSourceCount()) ).
#else,#end#end
#end


%%%%%% RESERVED **********************
% RESERVED (if multiple sources, and−style)
#foreach( $r in $program.getModel().relations( "reserved" ) )
#foreach( $s in $r.getSources() )
reserved($s.getId(),$r.getTarget().getId()).
#end
#end

%%%%%% SOCIAL holds **********************
#foreacheach( $r in $program.getModel().relations( "wanted" ) )
% $r.getTarget().getId() wants the following goals $r.getSources()
ftx($r.getTarget().getId()) :− #foreach( $s in $r.getSources() )st($s.getId()),#end ftx($r.getTarget().getId()cop).
fux($r.getTarget().getId()) :− not ftx($r.getTarget().getId()), fux($r.getTarget().getId()cop).
ffx($r.getTarget().getId()) :− not ftx($r.getTarget().getId()), not fux($r.getTarget().getId()cop).
#foreach( $s in $r.getSources() )wanted($s.getId(),$r.getTarget().getId()).#end
#end


%%%%%% Srole coplay Lroles **********************
#foreach( $r in $program.getModel().relations( "coplay" ) )
% the social role $r.getTarget().getId() coplays the following legal roles: $r.getSources()
ftx($r.getTarget().getId()cop) :− #foreach( $s in $r.getSources() )ftx($s.getId())#if(($r.sources.indexOf($s)+1) == (
    $r.getSourceCount()) ).
#else,#end#end
```

```
#foreach( $s in $r.getSources() )
ffx($r.getTarget().getId()cop) :- ffx($s.getId()).
#end
fux($r.getTarget().getId()cop) :- not ftx($r.getTarget().getId()cop), not ffx($r.getTarget().getId()cop).
#end
######## add fact coplay exists
#foreach( $r in $program.getModel().relations( "coplay" ) )
#foreach( $s in $r.getSources() )
coplay($s.getId(),$r.getTarget().getId()).
#end
#end

############## RESERVE SCHEMA
#foreach( $r in $program.getModel().relations( "coplay" ) )
#foreach( $s in $r.getSources() )
#foreach( $b in $program.getModel().relations( "reserved" ) )
#if( $b.getTarget().getId()==$s.getId())
% if the coplayed LRole $b.getTarget().getId() has some reserved situations
% then for each situation $b.getSources() that is reserved for $b.getTarget().getId(), i check the patterns are ok
#foreach( $w in $b.getSources())
schemadue($b.getTarget().getId(),$r.getTarget().getId(),$w.getId()) :- coplay($b.getTarget().getId(),$r.getTarget().
      getId()), reserved($w.getId(),$b.getTarget().getId()), reserved($w.getId(),$r.getTarget().getId()).
schemaduea($b.getTarget().getId(),$r.getTarget().getId(),$w.getId()) :- reserved($w.getId(),$b.getTarget().getId()),
      reserved($w.getId(),$r.getTarget().getId()).
schemadueb($b.getTarget().getId(),$r.getTarget().getId(),$w.getId()) :- coplay($b.getTarget().getId(),$r.getTarget().
      getId()), reserved($w.getId(),$b.getTarget().getId()).
#foreach( $hold in $program.getModel().relations( "hold" ) )
#foreach( $norm in $hold.getSources())
%% VIO2b:
viodueb($norm.getId()) :- not schemadue($b.getTarget().getId(), $r.getTarget().getId(), $w.getId()), schemadueb($b.
      getTarget().getId(),$r.getTarget().getId(),$w.getId()).
%% VIO2a:
vioduea($norm.getId()) :- not schemadue($b.getTarget().getId(), $r.getTarget().getId(), $w.getId()), schemaduea($b.
      getTarget().getId(),$r.getTarget().getId(),$w.getId()).
#foreach( $goal in $program.getModel().propositions( "goal" ) )
schematre($w.getId(),$goal.getId(),$r.getTarget().getId()) :- reserved($w.getId(),$r.getTarget().getId()), wanted(
      $goal.getId(),$r.getTarget().getId()), satisfy($w.getId(),$goal.getId()).
schematrea($w.getId(),$goal.getId(),$r.getTarget().getId()) :- reserved($w.getId(),$r.getTarget().getId()), wanted(
      $goal.getId(),$r.getTarget().getId()).
schematreb($w.getId(),$goal.getId(),$r.getTarget().getId()) :- reserved($w.getId(),$r.getTarget().getId()), satisfy(
      $w.getId(),$goal.getId()).
%% vio3a
viotrea($norm.getId()) :- not schematre($w.getId(),$goal.getId(),$r.getTarget().getId()), schematrea($w.getId(),$goal.
      getId(),$r.getTarget().getId()).
%% vio3b
viotreb($norm.getId()) :- not schematre($w.getId(),$goal.getId(),$r.getTarget().getId()), schematreb($w.getId(),$goal.
      getId(),$r.getTarget().getId()).
#end#end#end
#end#end#end
#end
#end


% Scenario SITUAZIONI
```

```
#foreach( $s in $program.getModel().propositions( "situation" ) )
st($s.getId()) v sf($s.getId()) v su($s.getId()).
#end
```

Bibliography

Abiteboul, Serge; Hull, Richard, and Vianu, Victor. *Foundations of Databases*. Addison-Wesley, 1995. ISBN 0-201-53771-0.

Alviano et al., Mario. The disjunctive datalog system dlv. In *Datalog*, pages 282–301, 2010.

Amyot, Daniel. Introduction to the user requirements notation: learning by example. *Computer Networks*, 42(3): 285–301, 2003. doi: http://dx.doi.org/10.1016/S1389-1286(03)00244-5. URL http://dx.doi.org/10.1016/S1389-1286(03)00244-5.

Amyot, Daniel; Horkoff, Jennifer; Gross, Daniel, and Mussbacher, Gunter. A lightweight GRL profile for i* modeling. In *Advances in Conceptual Modeling - Challenging Perspectives, ER 2009 Workshops CoMoL, ETheCoM, FP-UML, MOST-ONISW, QoIS, RIGiM, SeCoGIS, Gramado, Brazil, November 9-12, 2009. Proceedings*, pages 254–264, 2009. doi: http://dx.doi.org/10.1007/978-3-642-04947-7_31. URL http://dx.doi.org/10.1007/978-3-642-04947-7_31.

Amyot, Daniel; Ghanavati, Sepideh; Horkoff, Jennifer; Mussbacher, Gunter; Peyton, Liam, and Yu, Eric. Evaluating goal models within the goal-oriented requirement language. *International Journal of Intelligent Systems*, 25(8):841–877, 2010. ISSN 1098-111X. doi: http://dx.doi.org/10.1002/int.20433.

Antón, Annie I. and Earp, Julia Brande. A requirements taxonomy for reducing web site privacy vulnerabilities. *Requir. Eng.*, 9(3):169–185, 2004. doi: 10.1007/s00766-003-0183-z. URL http://dx.doi.org/10.1007/s00766-003-0183-z.

Antón, Annie I. and Potts, Colin. The use of goals to surface requirements for evolving systems. In *Proceedings of the 20th International Conference on Software Engineering*, ICSE '98, pages 157–166, 1998.

Antón, Annie I.; Earp, Julia Brande, and Reese, Angela. Analyzing website privacy requirements using a privacy goal taxonomy. In *10th Anniversary IEEE Joint International Conference on Requirements Engineering (RE 2002), 9-13 September 2002, Essen, Germany*, pages 23–31, 2002. doi: 10.1109/ICRE.2002.1048502. URL http://dx.doi.org/10.1109/ICRE.2002.1048502.

Antonini, Alessio; Boella, Guido; Hulstijn, Joris, and Humphreys, Llio. Requirements of legal knowledge management systems to aid normative reasoning in specialist domains. In *New Frontiers in Artificial Intelligence - JSAI-isAI 2013 Workshops, LENLS, JURISIN, MiMI, AAA, and DDS, Kanagawa, Japan, October 27-28, 2013, Revised Selected Papers*, pages 167–182, 2013. doi: 10.1007/978-3-319-10061-6_12. URL http://dx.doi.org/10.1007/978-3-319-10061-6_12.

Antoniou, Grigoris; Billington, David, and Maher, Michael J. On the analysis of regulations using defeasible rules. In *HICSS*, 1999. doi: http://computer.org/proceedings/hicss/0001/00016/00016033abs.htm.

Asaro, Carmelo; Biasiotti, Maria Angela; Guidotti, Paolo; Papini, Maurizio; Sagri, Maria-Teresa; Tiscornia, Daniela, and Court, Lucca. A Domain Ontology: Italian Crime Ontology. In *Proceedings of the ICAIL 2003 Workshop on Legal Ontologies and Web based legal information management*, pages 1–7, 2003.

Asnar, Yudistira and Massacci, Fabio. A method for security governance, risk, and compliance (GRC): A goal-process approach. In *Foundations of Security Analysis and Design VI - FOSAD Tutorial Lectures*, pages 152–184, 2011. doi: 10.1007/978-3-642-23082-0_6. URL http://dx.doi.org/10.1007/978-3-642-23082-0_6.

Awad, Ahmed; Decker, Gero, and Weske, Mathias. Efficient compliance checking using BPMN-Q and temporal logic. In *Business Process Management, 6th International Conference, BPM 2008, Milan, Italy, September 2-4, 2008. Proceedings*, pages 326–341, 2008. doi: 10.1007/978-3-540-85758-7_24. URL http://dx.doi.org/10.1007/978-3-540-85758-7_24.

Banescu, Sebastian; Petkovic, Milan, and Zannone, Nicola. Measuring privacy compliance using fitness metrics. In *Business Process Management - 10th International Conference, BPM 2012, Tallinn, Estonia, September 3-6, 2012. Proceedings*, pages 114–119, 2012. doi: 10.1007/978-3-642-32885-5_8. URL http://dx.doi.org/10.1007/978-3-642-32885-5_8.

Barth, A.; Datta, A.; Mitchell, J.C., and Nissenbaum, H. Privacy and contextual integrity: framework and applications. In *Security and Privacy, 2006 IEEE Symposium on*, pages 15 pp.–198, May 2006. doi: 10.1109/SP.2006.32.

Becker, Jörg; Delfmann, Patrick; Eggert, Mathias, and Schwittay, Sebastian. Generalizability and applicability of model-based business process compliance-checking approaches âĂŤ a state-of-the-art analysis and research roadmap. *BuR - Business Research*, 5(2):221–247, 2012. ISSN 2198-3402. doi: 10.1007/BF03342739. URL http://dx.doi.org/10.1007/BF03342739.

Bench-Capon, Trevor J. M.; Araszkiewicz, Michal; Ashley, Kevin D.; Atkinson, Katie; Bex, Floris; Borges, Filipe; Bourcier, Danièle; Bourgine, Paul; Conrad, Jack G.; Francesconi, Enrico; Gordon, Thomas F.; Governatori, Guido; Leidner, Jochen L.; Lewis, David D.; Loui, Ronald Prescott; McCarty, L. Thorne; Prakken, Henry; Schilder, Frank; Schweighofer, Erich; Thompson, Paul; Tyrrell, Alex; Verheij, Bart; Walton, Douglas N., and Wyner, Adam Zachary. A history of ai and law in 50 papers: 25 years of the international conference on ai and law. *Artif. Intell. Law*, 20(3):215–319, 2012. doi: http://dx.doi.org/10.1007/s10506-012-9131-x.

Bench-Capon et al., Trevor. A history of AI and Law in 50 papers: 25 years of the international conference on AI and Law. *Artificial Intelligence and Law*, pages 215–319, September 2012. ISSN 0924-8463. doi: 10.1007/s10506-012-9131-x. URL http://www.springerlink.com/index/10.1007/s10506-012-9131-x.

Boella, Guido and Torre, Leendert Van Der. Introduction to normative multiagent systems. *Computational and Mathematical Organization Theory*, 12:71–79, 2006.

Boella, Guido; Governatori, Guido; Rotolo, Antonino, and van der Torre, Leendert W. N. *Lex Minus Dixit Quam Voluit, Lex Magis Dixit Quam Voluit*: A formal study on legal compliance and interpretation. In *AI Approaches to the Complexity of Legal Systems. Complex Systems, the Semantic Web, Ontologies, Argumentation, and Dialogue (AICOL-II/JURIX 2009)*, pages 162–183, 2009. doi: 10.1007/978-3-642-16524-5_11.

Boella, Guido; Humphreys, Llio; Martin, Marco; Rossi, Piercarlo, and van der Torre, Leendert W. N. Eunomos, a legal document and knowledge management system to build legal services. In *AI Approaches to the Complexity of Legal Systems. Models and Ethical Challenges for Legal Systems, Legal Language and Legal Ontologies, Argumentation and Software Agents - International Workshop AICOL-III, Held as Part of the 25th IVR Congress, Frankfurt am Main, Germany, August 15-16, 2011. Revised Selected Papers*, pages 131–146, 2011. doi: 10.1007/978-3-642-35731-2_9. URL http://dx.doi.org/10.1007/978-3-642-35731-2_9.

Breaux, Travis D.; Vail, Matthew W, and Antón, Annie I. Towards Regulatory Compliance: Extracting Rights and Obligations to Align Requirements with Regulations. In *14th IEEE International Requirements Engineering Conference (RE'06)*, 2006. ISBN 0769525555.

Breaux, Travis D.; Antón, Annie I., and Spafford, Eugene H. A distributed requirements management framework for legal compliance and accountability. *Computers & Security*, 28(1-2):8–17, February 2009. ISSN 01674048. doi: 10.1016/j.cose.2008.08.001. URL http://linkinghub.elsevier.com/retrieve/pii/S0167404808000679.

Bresciani, Paolo; Perini, Anna; Giorgini, Paolo; Giunchiglia, Fausto, and Mylopoulos, John. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, May 2004. doi: 10.1023/B:AGNT.0000018806.20944.ef.

Butler, Tom and McGovern, Damien. A conceptual model and IS framework for the design and adoption of environmental compliance management systems - for special issue on governance, risk and compliance in IS. *Information Systems Frontiers*, 14(2):221–235, 2012. doi: 10.1007/s10796-009-9197-5. URL http://dx.doi.org/10.1007/s10796-009-9197-5.

Casellas, Nuria. *Legal Ongology Engineering*. Springer, law, gover edition, 2011. ISBN 978-94-007-1496-0. URL http://www.springer.com/law/book/978-94-007-1496-0.

Cheng, Betty H. C. and others, . Software Engineering for Self-Adaptive Systems: A Research Roadmap. In *Software Engineering for Self-Adaptive Systems*, pages 1–26. Springer, 2009.

Ciaghi, Aaron and Villafiorita, Adolfo. Improving public administrations via law modeling and bpr. In *E-Infrastuctures and E-Services for Developing Countries*, volume 64 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 69–78. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-23828-4.

Compagna, Luca; El Khoury, Paul; Krausová, Alžběta; Massacci, Fabio, and Zannone, Nicola. How to integrate legal requirements into a requirements engineering methodology for the development of security and privacy patterns. *Artificial Intelligence and Law*, 17(1):1–30, November 2008. ISSN 0924-8463. doi: 10.1007/s10506-008-9067-3. URL http://www.springerlink.com/index/10.1007/s10506-008-9067-3.

Dardenne, Anne; van Lamsweerde, Axel, and Fickas, Stephen. Goal-directed requirements acquisition. *Sci. Comput. Program.*, 20(1-2):3–50, Apr 1993. doi: 10.1016/0167-6423(93)90021-G.

Dignum, Frank. Autonomous agents with norms. *Artificial Intelligence and Law*, 7:69–79, 1999.

Fellmann, Michael and Zasada, Andrea. State-of-the-art of business process compliance approaches. In *22st European Conference on Information Systems, ECIS 2014, Tel Aviv, Israel, June 9-11, 2014*, 2014. URL http://aisel.aisnet.org/ecis2014/proceedings/track06/8.

Fikes, Richard E. and Nilsson, Nils J. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208, 1971. ISSN 0004-3702. doi: http://dx.doi.org/10.1016/0004-3702(71)90010-5. URL http://www.sciencedirect.com/science/article/pii/0004370271900105.

Gangemi, Aldo; Sagri, Maria-Teresa, and Tiscornia, Daniela. A constructive framework for legal ontologies. In *Law and the Semantic Web: Legal Ontologies, Methodologies, Legal Information Retrieval, and Applications [outcome of the Workshop on Legal Ontologies and Web-Based Legal Information Management, June 28, 2003, Edinburgh, UK & International Seminar on Law and the Semantic Web, November 20-21, 2003, Barcelona, Spain]*, pages 97–124, 2003. doi: 10.1007/978-3-540-32253-5_7. URL http://dx.doi.org/10.1007/978-3-540-32253-5_7.

García, Roberto and Gil, Rosa. Copyright licenses reasoning an OWL-DL ontology. In *Law, Ontologies and the Semantic Web - Channelling the Legal Information Flood*, pages 145–162, 2009. doi: 10.3233/978-1-58603-942-4-145. URL http://dx.doi.org/10.3233/978-1-58603-942-4-145.

Ghanavati, Sepideh; Amyot, Daniel, and Peyton, Liam. Towards a framework for tracking legal compliance in healthcare. In *CAiSE*, pages 218–232, 2007.

Ghanavati, Sepideh; Amyot, Daniel, and Peyton, Liam. Compliance analysis based on a goal-oriented requirement language evaluation methodology. In *RE*, pages 133–142, 2009.

Ghanavati, Sepideh; Amyot, Daniel, and Rifaut, André. Legal goal-oriented requirement language (legal grl) for modeling regulations. In *MiSE*, pages 1–6, 2014a.

Ghanavati, Sepideh; Ingolfo, Silvia, and Siena, Alberto. Exploring legal business process paths. In *The Practice of Enterprise Modeling - 7th IFIP WG 8.1 Working Conference, PoEM 2014, Manchester, UK, November 12-13, 2014. Proceedings*, pages 1–10, 2014b. doi: 10.1007/978-3-662-45501-2_1. URL http://dx.doi.org/10.1007/978-3-662-45501-2_1.

Giorgini, Paolo; Massacci, Fabio; Mylopoulos, John, and Zannone, Nicola. Requirements engineering meets trust management: Model, methodology, and reasoning. In *Trust Management, Second International Conference, iTrust 2004, Oxford, UK, March 29 - April 1, 2004, Proceedings*, pages 176–190, 2004. doi: 10.1007/978-3-540-24747-0_14. URL http://dx.doi.org/10.1007/978-3-540-24747-0_14.

Giorgini, Paolo; Massacci, Fabio; Mylopoulos, John, and Zannone, Nicola. Modeling security requirements through ownership, permission and delegation. In *RE 2005*, 2005a.

Giorgini, Paolo; Mylopoulos, John, and Sebastiani, Roberto. Goal-oriented requirements analysis and reasoning in the tropos methodology. *Eng. Appl. of AI*, 18(2):159–171, 2005b. doi: http://dx.doi.org/10.1016/j.engappai.2004.11.017.

Gordon, David G. and Breaux, Travis D. Reconciling multi-jurisdictional legal requirements: A case study in requirements water marking. In *RE*, pages 91–100, 2012.

Gordon, David G. and Breaux, Travis D. Assessing regulatory change through legal requirements coverage modeling. In *RE*, pages 145–154, 2013.

Gordon, Thomas F. and Karacapilidis, Nikos. The zeno argumentation framework. In *Proceedings of the 6th international conference on Artificial intelligence and law*, ICAIL '97, pages 10–18, New York, NY, USA, 1997. ACM. ISBN 0-89791-924-6. doi: 10.1145/261618.261622. URL http://doi.acm.org/10.1145/261618.261622.

Governatori, Guido. Representing business contracts in ruleml. *International Journal of Cooperative Information Systems*, 14(02n03):181–216, 2005. doi: 10.1142/S0218843005001092.

Governatori, Guido and Rotolo, Antonino. How Do Agents Comply with Norms? *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, pages 488–491, 2009. doi: 10.1109/WI-IAT.2009.332. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5285032.

Hage, Jaap and Verheij, Bart. The law as a dynamic interconnected system of states of affairs: a legal top ontology. *Int. J. Hum.-Comput. Stud.*, 51(6):1043–1077, 1999. doi: 10.1006/ijhc.1999.0297. URL http://dx.doi.org/10.1006/ijhc.1999.0297.

Hansen, Jörg; Pigozzi, Gabriella, and van der Torre, L.W.N. Ten philosophical problems in deontic logic. In *Normative Multi-agent Systems*, 2007. doi: http://drops.dagstuhl.de/opus/volltexte/2007/941.

Hevner, Alan R.; March, Salvatore T.; Park, Jinsoo, and Ram, Sudha. Design science in information systems research. *MIS Q.*, 28(1):75–105, 2004.

Hoekstra, Rinke; Breuker, Joost; Bello, Marcello Di, and Boer, Alexander. The LKIF core ontology of basic legal concepts. In *Proceedings of the 2nd Workshop on Legal Ontologies and Artificial Intelligence Techniques June 4th, 2007, Stanford University, Stanford, CA, USA*, pages 43–63, 2007. URL http://ceur-ws.org/Vol-321/paper3.pdf.

Hohfeld, Wesley Newcomb. Fundamental legal conceptions as applied in judicial reasoning. *The Yale Law Journal*, 26:710–770, 1917. http://www.jstor.org/stable/786270.

Horkoff, Jennifer and Yu, Eric S. K. Finding solutions in goal models: An interactive backward reasoning approach. In *Conceptual Modeling - ER 2010, 29th International Conference on Conceptual Modeling, Vancouver, BC, Canada, November 1-4, 2010. Proceedings*, pages 59–75, 2010. doi: 10.1007/978-3-642-16373-9_5.

Horkoff, Jennifer and Yu, Eric S. K. Comparison and evaluation of goal-oriented satisfaction analysis techniques. *Requir. Eng.*, 18(3):199–222, 2013. doi: 10.1007/s00766-011-0143-y. URL http://dx.doi.org/10.1007/s00766-011-0143-y.

Horkoff, Jennifer; Borgida, Alexander; Mylopoulos, John; Barone, Daniele; Jiang, Lei; Yu, Eric S. K., and Amyot, Daniel. Making data meaningful: The business intelligence model and its formal semantics in description logics. In *On the Move to Meaningful Internet Systems: OTM 2012, Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2012, Rome, Italy, September 10-14, 2012. Proceedings, Part II*, pages 700–717, 2012. doi: http://dx.doi.org/10.1007/978-3-642-33615-7_17. URL http://dx.doi.org/10.1007/978-3-642-33615-7_17.

Horkoff, Jennifer; Barone, Daniele; Jiang, Lei; Yu, Eric S. K.; Amyot, Daniel; Borgida, Alexander, and Mylopoulos, John. Strategic business modeling: representation and reasoning. *Software and System Modeling*, 13: 1015–1041, 2014. doi: 10.1007/s10270-012-0290-8. URL http://dx.doi.org/10.1007/s10270-012-0290-8.

in Software Engineering. Conceptual Modeling Foundations, Non-Functional Requirements and Applications, . *Requirements Engineering: From System Goals to UML Models to Software Specifications*, volume 5, pages 363–379. Kluwer Academic Publishing, 2000.

Ingolfo, Silvia and Souza, Vítor Estêvão Silva. Law and adaptivity in requirements engineering. In *Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2013, San Francisco, CA, USA, May 20-21, 2013*, pages 163–168, 2013. doi: 10.1109/SEAMS.2013.6595503. URL http://dx.doi.org/10.1109/SEAMS.2013.6595503.

Ingolfo, Silvia; Siena, Alberto; Mylopoulos, John; Susi, Angelo, and Perini, Anna. Arguing regulatory compliance of software requirements. *Accepted for publication in Journal of Data & Knowledge Engineering (DKE)*, 2012.

Ingolfo, Silvia; Siena, Alberto; Jureta, Ivan; Susi, Angelo; Perini, Anna, and Mylopoulos, John. Choosing compliance solutions through stakeholder preferences. In *REFSQ 2013*, volume 7830 of *Lecture Notes in Computer Science*, pages 206–220. 2013. URL http://dx.doi.org/10.1007/978-3-642-24606-7_5.

Johnston, Benjamin and Governatori, Guido. Induction of defeasible logic theories in the legal domain. In *ICAIL*, pages 204–213, 2003.

Jureta, Ivan; Borgida, Alexander; Ernst, Neil A., and Mylopoulos, John. Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling. In *RE 2010, 18th IEEE International Requirements Engineering Conference, Sydney, New South Wales, Australia, September 27 - October 1, 2010*, pages 115–124, 2010. doi: http://dx.doi.org/10.1109/RE.2010.24. URL http://dx.doi.org/10.1109/RE.2010.24.

Khadraoui, Abdelaziz; Leonard, Michel; Thi, Thanh Thoa Pham, and Helfert, Markus. A Framework for Compliance of Legacy Information Systems with Legal Aspect. In *AIS Trans. Enterprise Sys.* GITO Publishing GmbH, 2009.

Kharbili, Marwane El; Ma, Qin; Kelsen, Pierre, and Pulvermüller, Elke. Enterprise regulatory compliance modeling using corel: An illustrative example. In *13th IEEE Conference on Commerce and Enterprise Computing, CEC 2011, Luxembourg-Kirchberg, Luxembourg, September 5-7, 2011*, pages 185–190, 2011. doi: 10.1109/CEC.2011.39. URL http://dx.doi.org/10.1109/CEC.2011.39.

Kiyavitskaya, Nadzeya; Zeni, Nicola; Breaux, Travis D.; Antón, Annie I.; Cordy, James R.; Mich, Luisa, and Mylopoulos, John. Automating the extraction of rights and obligations for regulatory compliance. *... Modeling-ER 2008*, pages 1–14, 2008. URL http://www.springerlink.com/index/dn7240937t72lx46.pdf.

Knuplesch, David; Ly, LinhThao; Rinderle-Ma, Stefanie; Pfeifer, Holger, and Dadam, Peter. On enabling data-aware compliance checking of business process models. In Parsons, Jeffrey; Saeki, Motoshi; Shoval, Peretz; Woo, Carson, and Wand, Yair, editors, *Conceptual Modeling âĂŞ ER 2010*, volume 6412 of *Lecture Notes in Computer Science*, pages 332–346. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-16372-2. doi: 10.1007/978-3-642-16373-9_24. URL http://dx.doi.org/10.1007/978-3-642-16373-9_24.

Leone, Nicola; Pfeifer, Gerald; Faber, Wolfgang; Eiter, Thomas; Gottlob, Georg; Perri, Simona, and Scarcello, Francesco. The dlv system for knowledge representation and reasoning. *ACM Trans. Comput. Log.*, 7(3): 499–562, 2006.

Liu, Ying; Müller, Samuel, and Xu, Ke. A static compliance-checking framework for business process models. *IBM Systems Journal*, 46(2):335–362, 2007. doi: 10.1147/sj.462.0335. URL http://dx.doi.org/10.1147/sj.462.0335.

Massacci, Fabio; Prest, Marco, and Zannone, Nicola. Using a security requirements engineering methodology in practice: The compliance with the italian data protection legislation. *Computer Standards & Interfaces*, 27(5): 445–455, 2005. doi: 10.1016/j.csi.2005.01.003. URL http://dx.doi.org/10.1016/j.csi.2005.01.003.

Massey, Aaron K.; Smith, Ben; Otto, Paul N., and Antón, Annie I. Assessing the accuracy of legal implementation readiness decisions. In *RE 2011, 19th IEEE International Requirements Engineering Conference, Trento, Italy, August 29 2011 - September 2, 2011*, pages 207–216, 2011. doi: 10.1109/RE.2011.6051661. URL http://dx.doi.org/10.1109/RE.2011.6051661.

Massey, Aaron K.; Eisenstein, Jacob; Antón, Annie I., and Swire, Peter P. Automated text mining for requirements analysis of policy documents. In *RE*, pages 4–13, 2013.

Massey, Aaron K.; Rutledge, Richard L.; Antón, Annie I., and Swire, Peter P. Identifying and classifying ambiguity for regulatory requirements. In *IEEE 22nd International Requirements Engineering Conference, RE 2014, Karlskrona, Sweden, August 25-29, 2014*, pages 83–92, 2014. doi: 10.1109/RE.2014.6912250. URL http://dx.doi.org/10.1109/RE.2014.6912250.

Maxwell, Jeremy C. and Anton, Annie I. Developing Production Rule Models to Aid in Acquiring Requirements from Legal Texts. *2009 17th IEEE International Requirements Engineering Conference*, pages 101–110, August 2009. doi: 10.1109/RE.2009.21. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5328596.

Maxwell, Jeremy C.; Antón, Annie I., and Swire, Peter P. A legal cross-references taxonomy for identifying conflicting software requirements. In *RE*, pages 197–206, 2011.

Maxwell, Jeremy C; Anton, Annie I; Swire, Peter; Engineering, College, and Carolina, North. Managing Changing Compliance Requirements by Predicting Regulatory Evolution An Adaptability Framework. In *2012 20th IEEE International Requirements Engineering Conference*, pages 101–110, 2012a. ISBN 9781467327848.

Maxwell, Jeremy C.; Antón, Annie I., and Swire, Peter P. Managing changing compliance requirements by predicting regulatory evolution. In *RE*, pages 101–110, 2012b.

Stanford Encyclopedia, . Propositions. http://plato.stanford.edu/entries/propositions/, 2005. Accessed May 2015.

McCarty, L. Thorne. A language for legal discourse i: Basic features. In *ICAIL*, pages 180–189, 1989. doi: http://doi.acm.org/10.1145/74014.74037.

Mylopoulos, J.; Chung, L., and Nixon, B. Representing and using nonfunctional requirements: a process-oriented approach. *Software Engineering, IEEE Transactions on*, 18(6):483–497, Jun 1992. doi: 10.1109/32.142871.

Nilsson, Nils J. *Problem-Solving Methods in Artificial Intelligence*. McGraw-Hill Pub. Co., 1971. ISBN 0070465738.

Patel-Schneider, Peter F. and Sebastiani, Roberto. A new general method to generate random modal formulae for testing decision procedures. *J. Artif. Intell. Res. (JAIR)*, 18:351–389, 2003.

Rifaut, André and Dubois, Eric. Using goal-oriented requirements engineering for improving the quality of iso/iec 15504 based compliance assessment frameworks. In *RE*, pages 33–42, 2008.

Rodríguez-Doncel, Víctor; Santos, Cristiana, and Casanovas, Pompeu. Ontology-driven legal support-system in the air transport passenger domain. In *Proceedings of the Semantic Web for the Law and Second Doctoral Consortium Workshops Co-located with 27th International Conference on Legal Knowledge and Information Systems, Krakow, Poland, December 10th-12th, 2014.*, 2014. URL http://ceur-ws.org/Vol-1296/paper3.pdf.

Rubino, Rossella; Rotolo, Antonino, and Sartor, Giovanni. An OWL ontology of fundamental legal concepts. In *Legal Knowledge and Information Systems - JURIX 2006: The Nineteenth Annual Conference on Legal Knowledge and Information Systems, Paris, France, 7-9 December 2006*, pages 101–110, 2006. URL http://www.booksonline.iospress.nl/Content/View.aspx?piid=2372.

Sadiq, Shazia Wasim; Governatori, Guido, and Namiri, Kioumars. Modeling control objectives for business process compliance. In *Business Process Management, 5th International Conference, BPM 2007, Brisbane, Australia, September 24-28, 2007, Proceedings*, pages 149–164, 2007. doi: 10.1007/978-3-540-75183-0_12. URL http://dx.doi.org/10.1007/978-3-540-75183-0_12.

Schultz, Martin. Towards an empirically grounded conceptual model for business process compliance. In *Conceptual Modeling - 32th International Conference, ER 2013, Hong-Kong, China, November 11-13, 2013. Proceedings*, pages 138–145, 2013. doi: 10.1007/978-3-642-41924-9_13. URL http://dx.doi.org/10.1007/978-3-642-41924-9_13.

Sebastiani, Roberto; Giorgini, Paolo, and Mylopoulos, John. Simple and minimum-cost satisfiability for goal models. In *Advanced Information Systems Engineering, 16th International Conference, CAiSE 2004, Riga, Latvia, June 7-11, 2004, Proceedings*, pages 20–35, 2004. doi: http://dx.doi.org/10.1007/978-3-540-25975-6_4. URL http://dx.doi.org/10.1007/978-3-540-25975-6_4.

Sen, S.; Guha, S.; Datta, A.; Rajamani, S.K.; Tsai, J., and Wing, J.M. Bootstrapping privacy compliance in big data systems. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 327–342, May 2014. doi: 10.1109/SP.2014.28.

Shamsaei, Azalia. Indicator-based policy compliance of business processes. In *Proceedings of the CAiSE Doctoral Consortium 2011, London, United Kingdom, June 21, 2011*, pages 3–14, 2011. URL http://ceur-ws.org/Vol-731/02.pdf.

Shamsaei, Azalia; Pourshahid, Alireza, and Amyot, Daniel. Business process compliance tracking using key performance indicators. In *Business Process Management Workshops - BPM 2010 International Workshops and Education Track, Hoboken, NJ, USA, September 13-15, 2010, Revised Selected Papers*, pages 73–84, 2010. doi: 10.1007/978-3-642-20511-8_7. URL http://dx.doi.org/10.1007/978-3-642-20511-8_7.

Siena, A. *Engineering law-compliant requirements. The Nòmos framework.* PhD Thesis, University of Trento, Italy, 2010.

Siena, Alberto; Mylopoulos, John; Perini, Anna, and Susi, Angelo. Designing law-compliant software requirements. In *Proceedings of the 28th International Conference on Conceptual Modeling*, ER'09, pages 472–486, 2009. doi: 10.1007/978-3-642-04840-1_35.

Singh, Munindar P. An ontology for commitments in multiagent systems. *Artif. Intell. Law*, 7(1):97–113, 1999.

Singh, Munindar P. Norms as a basis for governing sociotechnical systems. *ACM TIST*, 5(1):21, 2013.

Souza, Vítor E. S.; Lapouchnian, Alexei; Robinson, William N., and Mylopoulos, John. Awareness Requirements for Adaptive Systems. In *Proc. of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 60–69. ACM, 2011.

Souza, Vítor E. S.; Lapouchnian, Alexei, and Mylopoulos, John. (Requirement) Evolution Requirements for Adaptive Systems. In *Proc. of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 155–164. IEEE, 2012.

Tawhid, Rasha; Braun, Edna; Cartwright, Nick; Alhaj, Mohammad; Mussbacher, Gunter; Shamsaei, Azalia; Amyot, Daniel; Behnam, Saeed Ahmadi, and Richards, Gregory. Towards outcome-based regulatory compliance in aviation security. In *RE*, pages 267–272, 2012.

Valente, André; Breuker, Joost, and Brouwer, Bob. Legal modeling and automated reasoning with ON-LINE. *Int. J. Hum.-Comput. Stud.*, 51(6):1079–1125, 1999. doi: 10.1006/ijhc.1999.0298. URL http://dx.doi.org/10.1006/ijhc.1999.0298.

van Engers, Tom M. and Nijssen, Sjir. Connecting people: Semantic-conceptual modeling for laws and regulations. In *Electronic Government - 13th IFIP WG 8.5 International Conference, EGOV 2014, Dublin, Ireland, September 1-3, 2014. Proceedings*, pages 133–146, 2014. doi: 10.1007/978-3-662-44426-9_11. URL http://dx.doi.org/10.1007/978-3-662-44426-9_11.

van Lamsweerde, A. Goal-oriented requirements engineering: a guided tour. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pages 249–262, 2001. doi: 10.1109/ISRE.2001.948567.

van Lamsweerde, A. Goal-oriented requirements enginering: a roundtrip from research to practice [enginering read engineering]. In *Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International*, pages 4–7, Sept 2004. doi: 10.1109/ICRE.2004.1335648.

van Lamsweerde, A. *Requirements Engineering: From System Goals to UML Models to Software Specifications.* Wiley, 2009. ISBN 978-0-470-01270-3.

van Lamsweerde, Axel and Letier, Emmanuel. Handling obstacles in goal-oriented requirements engineering. *IEEE Trans. Software Eng.*, 26(10):978–1005, 2000. doi: http://dx.doi.org/10.1109/32.879820. URL http://dx.doi.org/10.1109/32.879820.

van Lamsweerde, Axel; Darimont, Robert, and Letier, Emmanuel. Managing conflicts in goal-driven requirements engineering. *IEEE Trans. Software Eng.*, 24(11):908–926, 1998. doi: http://dx.doi.org/10.1109/32.730542. URL http://dx.doi.org/10.1109/32.730542.

Veeningen, Meilof; de Weger, Benne, and Zannone, Nicola. Formal privacy analysis of communication protocols for identity management. In *Information Systems Security - 7th International Conference, ICISS 2011, Kolkata, India, December 15-19, 2011, Procedings*, pages 235–249, 2011. doi: 10.1007/978-3-642-25560-1_16. URL http://dx.doi.org/10.1007/978-3-642-25560-1_16.

Wikipedia, . Norm (philosophy). http://en.wikipedia.org/wiki/Norm_(philosophy), 2005. Accessed May 2015.

Witt, Sören; Feja, Sven; Speck, Andreas, and Hadler, Christian. Business application modeler: A process model validation and verification tool. In *IEEE 22nd International Requirements Engineering Conference, RE 2014, Karlskrona, Sweden, August 25-29, 2014*, pages 333–334, 2014. doi: 10.1109/RE.2014.6912283. URL http://dx.doi.org/10.1109/RE.2014.6912283.

Young, Jessica D. Commitment analysis to operationalize software requirements from privacy policies. *Requirements Engineering*, 16(1):33–46, June 2010. ISSN 0947-3602. doi: 10.1007/s00766-010-0108-6. URL http://www.springerlink.com/index/10.1007/s00766-010-0108-6.

Yu, E.S.K. *Modelling Strategic Relationships for Process Reengineering.* PhD thesis, University of Toronto, Toronto, Ont., Canada, 1996.

Yu, E.S.K. Towards modelling and reasoning support for early-phase requirements engineering. In *Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on*, pages 226–235, 1997a. doi: 10.1109/ISRE.1997.566873.

Yu, E.S.K. Towards modelling and reasoning support for early-phase requirements engineering. In *Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on*, pages 226 –235, 1997b. http://dx.doi.org/10.1109/ISRE.1997.566873.

Zave, Pamela and Jackson, Michael. Four dark corners of requirements engineering. *ACM Trans. Softw. Eng. Methodol.*, 6(1):1–30, January 1997. ISSN 1049-331X. doi: 10.1145/237432.237434. URL http://doi.acm.org/10.1145/237432.237434.