Compliance by design: Synthesis of business processes by declarative specifications

FRANCESCO OLIVIERI
B.Sc. CS M.Sc.

Institute for Integrated and Intelligent Systems
Griffith Science Group
Griffith University

Submitted in fulfilment of the requirements of the degree of
Doctor of Philosophy

Under the Co-tutelle Agreement between Griffith University, Brisbane, Australia and the Department of Computer Science, Università degli Studi di Verona, Italy

May, 2014
UNIVERSITÀ DEGLI STUDI DI VERONA

DEPARTMENT OF
COMPUTER SCIENCE

GRADUATE SCHOOL OF
SCIENCES ENGINEERING MEDICINE

DOCTORAL PROGRAM IN
COMPUTER SCIENCE

WITH THE FINANCIAL CONTRIBUTION OF
NICTA QUEENSLAND, AUSTRALIA

Cycle XXV Year 2010

TITLE OF THE DOCTORAL THESIS

COMPLIANCE BY DESIGN:
SYNTHESIS OF BUSINESS PROCESSES
BY DECLARATIVE SPECIFICATIONS

IN CO-TUTELLE DE THÈSE WITH
GRIFFITH UNIVERSITY, BRISBANE, AUSTRALIA

S.S.D INF/01

SUPERVISORS: Dr. MATTEO CRISTANI, Università degli Studi di Verona, Italy
Prof. ABDUL SATTAR, Griffith University, Brisbane, Australia
Prof. GUIDO GOVERNATORI, NICTA, Brisbane, Australia

COORDINATOR: Prof. PAOLO FIORINI, Università degli Studi di Verona, Italy

Doctoral Student: FRANCESCO OLIVIERI

Signature: Francesco Olivieri
To myself. To Bilbo and Samwise.
(I’ve never liked Frodo that much)
Keywords
Agents, BDI Paradigm, Business Process, Business Process Compliance, Non-monotonic Reasoning, Modal Defeasible Logic, Graph Theory
Abstract

Business Process Compliance are three words which scholars use to describe what happens, or should happen, when two very different worlds collide. The first world is meant to represent enterprises and how they do what they do or, more simply, which procedures and processes they adopt to offer improved products to their customers. Scholars of the field refer to the Business Process Management as a “process optimisation process” and they study approaches, methodologies, and formal languages to describe and improve what they esteem as the heart of every organisation, the business process.

A business process can be visualised as a self-contained, temporal, and logical order in which a set of activities (tasks) are executed to achieve some business objectives. Within a business process, much information is available: The control flow describes what can be done and when, while the relevant data clarify what needs to be work on as well as which actors will do the work.

The second world is the world of governments, of consortia, of all those entities which have enough power to create regulations, norms, and policies which directly impact organisations. Such entities state the boundaries of legality by imposing which actions can be considered legal to be performed within the aforementioned business processes, and which actions should be avoided in order not to incur severe sanctions.

Business Process Compliance is the research field where scholars try to understand how organisations should behave in order to continue offering good products while respecting a set of regulations which strictly affect their processes. In general, a compliance regimen must include three interrelated but distinct perspectives of compliance: Corrective, detective, and preventative measures. While the first two measures try to mitigate or intervene after compliance breaches are detected, the preventative focus assumes a completely different perspective by stating that “compliance should be embedded into the business practice, rather than be seen as a distinct activity, (...) thus achieving compliance by design” [Sadiq and Governatori, 2015].

An issue of great importance is that of devising automated tools which are able to create an entirely new, compliant process starting from a fully declarative description of both the organisation and the environment it is acting in. Such a description include: (i) A set of business objectives to be reached, (ii) The specifications of a process, (iii) The norms ruling the organisational environment. This doctorate dissertation will confront this problem by following two sequential research paths.
First things first, we need a formalism able to state, given a particular context, which norms are in force, which objectives are desirable and which objectives are feasible, in order for the organisation to decide which objectives to commit to. Thus, we liken organisations to agents and, accordingly, refer to the literature of BDI (Belief-Desire-Intention) agents. The BDI architecture addresses how agents try to fulfil their goals based on the knowledge of the environment and a collection of plans. We shall analyse different notions of the concept of goal starting from the idea of sequences of “alternative acceptable outcomes”. We shall study the relationships between goals and concepts like agent’s beliefs, norms and desires, and propose a computationally oriented formalisation using a variant of Defeasible Logic extended with modal operators that will provide a suitable approach. The resulting system, being able to capture various nuances of the notion of goal, is of interest for business enterprises, for which the right decision is not only context-dependent but one which should be chosen from a preferably large pool of alternatives.

Finally, we shall propose algorithms to compute all provable and refutable conclusions of the modal defeasible theory, and we shall prove their soundness and computational complexity.

At the end of the aforementioned phase, the system (organisation) knows whether courses of action exist (in term of logical derivations) which lead to norm and outcome compliant situations. The second question then being how to determine which legitimate courses of action the agent may commit to, and how to transform them into a business process-like graphic notation.

We therefore shall propose algorithms which (i) Construct a graph by navigating backwards the derivation trees from the “compliant” outcomes up to the facts of the theory, and (ii) Transform such a graph by recognising JOIN and SPLIT patterns typical of process model notation. And, as stated before, we shall end the section by presenting a computational analysis.
Statement of Originality

This work has been submitted to both the Institute for Integrated and Intelligent Systems, Griffith Science Group - Griffith University and the Department of Computer Science - University of Verona in accordance to the joint doctorate program between the two universities. Other than that, this work has not previously been submitted for a degree or diploma in any other university. To the best of my knowledge and belief, the thesis contains no material previously published or written by a third person except where due reference is made in the thesis itself.

Doctoral Student: FRANCESCO OLIVIERI

Signature: Francesco Olivieri
Reddite quae sunt Caesaris Caesari et quae sunt Dei Deo. Id est, questo lavoro non avrebbe mai visto la luce senza (in ordine rigorosamente alfabetico): Guido, Matteo, Marianna, Simone. A loro va un particolare Grazie!, un bacio, un abbraccio e il mio affetto, che alla fine penso sia la cosa più importante, no..?

Ancora grazie a Simone, my buddy, mio fratello di vita. Sul serio, senza di te non ce l’avrei mai fatta. Mi hai supportato in tutto e per tutto. Mi hai sopportato anche laddove un altro mi “ci avrebbe mandato”. Con te ho pianto. Con te ho riso. Con te ho scalato. Con te ho placcato e mi sono sporcato di fango. Con te mi sono spaccato le ossa e sono andato all’ospedale. Con te sono diventato dottore. Tre volte, secondo alcuni. You are a great person, so remember: “Good things to good people”.

Poi, grazie papà Dino. Dopo tante litigiate non posso non ammettere che se sono arrivato fin qui, tanto lo devo a te e a come mi hai educato. Spero solo che un giorno tu possa capirlo e riconoscere i miei, forse piccoli, meriti.

Un grazie a mio fratello Luca. Se la piccola german sausage porta il tuo nome un motivo pur ci sarà. Perché il nostro legame sia sempre forte, non importa quanti oceani ci siano fra noi. Per me sei e sempre sarai, il mio fratellino.

Un grazie al mio ninja preferito. Tu mi hai insegnato cosa il significato di “amore incondizionato”. Grazie Carlo-sho (by Daddy-sampi).

Grazie al mio bundle of joy, german sausage Jabba Julian-san (Carlo ha deciso il -san). Il tuo sorriso da vampiro è il più bello che abbia mai visto (con quello di mamma s’intende).

Un grazie a Matteo. Un grande grazie. Enorme. Definirti semplicemente un mentore sarebbe non renderti giustizia. Tu sei sempre stato un modello per me. Lo so che lo sei per i tuoi figli. Tu sei un grand’uomo. Con una grande mente ma con un ancor più grande cuore.


Lastly, a special thanks to Mariana del alma mia. The Love of this life. And of all the lives we have spent together.
The last time I saw you we were just split in two...
# CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>iii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>v</td>
</tr>
<tr>
<td>List of Tables</td>
<td>vii</td>
</tr>
<tr>
<td>List of Algorithms</td>
<td>ix</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2 Concerning methodology</td>
<td>11</td>
</tr>
<tr>
<td>2.1 Of the defeasibility of knowledge: The case of Defeasible Logic</td>
<td>13</td>
</tr>
<tr>
<td>2.2 Concerning process graphs representation</td>
<td>18</td>
</tr>
<tr>
<td>2.2.1 Process graph notation</td>
<td>19</td>
</tr>
<tr>
<td>3 Towards a logical framework to model declarative specifications</td>
<td>23</td>
</tr>
<tr>
<td>3.1 Introducing Outcomes: A motivational detour</td>
<td>25</td>
</tr>
<tr>
<td>3.2 The intuition underneath the framework</td>
<td>28</td>
</tr>
<tr>
<td>3.2.1 Desires as acceptable outcomes</td>
<td>30</td>
</tr>
<tr>
<td>3.2.2 Goals as preferred outcomes</td>
<td>31</td>
</tr>
<tr>
<td>3.2.3 Two degrees of commitment: Intentions and social intentions</td>
<td>31</td>
</tr>
<tr>
<td>4 Of the formalisation of the logic</td>
<td>35</td>
</tr>
<tr>
<td>4.1 Language</td>
<td>36</td>
</tr>
<tr>
<td>4.1.1 Rule conversion</td>
<td>38</td>
</tr>
<tr>
<td>4.1.2 Conflict-detection/resolution</td>
<td>40</td>
</tr>
<tr>
<td>4.2 Inferential Mechanism</td>
<td>41</td>
</tr>
<tr>
<td>4.3 Norm and outcome compliance</td>
<td>50</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>4.4</td>
<td>Running examples - Part I</td>
</tr>
<tr>
<td>4.5</td>
<td>Theory extension calculus: Summary and related work</td>
</tr>
<tr>
<td>5</td>
<td>Of the algorithmic results for the logic extension computation</td>
</tr>
<tr>
<td>5.1</td>
<td>Notation for the algorithms</td>
</tr>
<tr>
<td>5.2</td>
<td>Algorithms for the logical extension calculus</td>
</tr>
<tr>
<td>5.3</td>
<td>Computational Results</td>
</tr>
<tr>
<td>6</td>
<td>Concerning process synthesis algorithms</td>
</tr>
<tr>
<td>6.1</td>
<td>Principles underlying our approach</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Algorithm notation</td>
</tr>
<tr>
<td>6.2</td>
<td>Algorithmic results for the process graph synthesis</td>
</tr>
<tr>
<td>6.3</td>
<td>Running examples - Part II</td>
</tr>
<tr>
<td>6.4</td>
<td>Computational results for synthesis algorithms</td>
</tr>
<tr>
<td>6.5</td>
<td>Summary and related work</td>
</tr>
<tr>
<td>7</td>
<td>There and back again: Conclusions and farewell</td>
</tr>
<tr>
<td>A</td>
<td>Proofs for the logic extension computation</td>
</tr>
<tr>
<td>B</td>
<td>Running example of Algorithm 6</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>2.1</td>
<td>BPMN-like representation of the logic.</td>
</tr>
<tr>
<td>2.2</td>
<td>Sequence of tasks</td>
</tr>
<tr>
<td>2.3</td>
<td>AND split/join gateways</td>
</tr>
<tr>
<td>2.4</td>
<td>XOR split/join gateways</td>
</tr>
<tr>
<td>2.5</td>
<td>OR split/join gateways</td>
</tr>
<tr>
<td>6.1</td>
<td>Instance of Split pattern recognition: Correct synthesis.</td>
</tr>
<tr>
<td>6.2</td>
<td>Two instances of Split pattern recognition: Wrong synthesis.</td>
</tr>
<tr>
<td>6.3</td>
<td>Process graph of theory $D_1$ resulting from Algorithm 4 ComplianceByDesign: Run of the algorithm up to Line 23</td>
</tr>
<tr>
<td>6.4</td>
<td>Process graph of theory $D$ resulting from Algorithm 4 ComplianceByDesign: Labelled edges and co-occurrence</td>
</tr>
<tr>
<td>6.5</td>
<td>Process graph of theory $D_2$ resulting from Algorithm 4 ComplianceByDesign: Run of the algorithm up to Line 32</td>
</tr>
<tr>
<td>6.6</td>
<td>Process graph of theory $D_2$ resulting from Algorithm 4 ComplianceByDesign: Final process graph.</td>
</tr>
<tr>
<td>6.7</td>
<td>PeoplEye’s process graph resulting from Algorithm 4 ComplianceByDesign: Run of the algorithm up to Line 32</td>
</tr>
<tr>
<td>6.8</td>
<td>PeoplEye’s process graph resulting from Algorithm 4 ComplianceByDesign: Graph with labelled edges.</td>
</tr>
<tr>
<td>6.9</td>
<td>Governatore’s process graph resulting from Algorithm 4 ComplianceByDesign: Run of the algorithm up to Line 32</td>
</tr>
<tr>
<td>6.10</td>
<td>Governatore’s process graph resulting from Algorithm 4 ComplianceByDesign: Co-occurrence of the tasks DOUGH and MOZZARELLA.</td>
</tr>
<tr>
<td>6.11</td>
<td>Governatore’s process graph resulting from Algorithm 4 ComplianceByDesign: final graph after the run of Algorithm 6.</td>
</tr>
</tbody>
</table>
B.1 Process graph of theory $D'$ resulting from Algorithm 4 COMPLIANCEByDesign: Run of the algorithm up to Line 33 . . . . . . 137
B.2 Process graph of theory $D'$ resulting from Algorithm 4 COMPLIANCEByDesign: Final process graph . . . . . . . . . . . . . . . . . . 138
LIST OF TABLES

4.1 Definition 11, Condition (2.3.2): Attacks and counterattacks . 47
LIST OF ALGORITHMS

<table>
<thead>
<tr>
<th></th>
<th>Algorithm</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DefeasibleExtension (Part 1)</td>
<td>71</td>
</tr>
<tr>
<td>1</td>
<td>DefeasibleExtension (Part 2)</td>
<td>72</td>
</tr>
<tr>
<td>2</td>
<td>Proved</td>
<td>74</td>
</tr>
<tr>
<td>3</td>
<td>Refuted</td>
<td>76</td>
</tr>
<tr>
<td>4</td>
<td>complianceByDesign (Part 1)</td>
<td>91</td>
</tr>
<tr>
<td>4</td>
<td>complianceByDesign (Part 2)</td>
<td>92</td>
</tr>
<tr>
<td>5</td>
<td>backTrack</td>
<td>93</td>
</tr>
<tr>
<td>6</td>
<td>splitPattern</td>
<td>97</td>
</tr>
</tbody>
</table>
The information available today on planet Earth is greater than what it was 1 year ago. The information available 1 year ago was greater than what it was 2 years ago. And if truth be told, surprisingly, the information available 2 years ago on planet Earth was greater than what it was 4 years ago, which in turn was sufficiently greater than what it was 8 years ago, which in turn was even greater than what it was 16 years ago. We can state that, almost certainly, the information available 16 years ago on planet Earth was much greater than what it was 32 years ago, which in turn was rather greater than what it was 64 years ago, when we had no CDs, no Apple, no Hepatitis B vaccine, no Microsoft Windows, no digital cellular phones either.

However, it is useless to say, that the information available 64 year ago on planet Earth was extraordinarily greater than what it was 128 years ago (just think that planet Earth of 128 years ago had not yet met “fellas” of the caliber of Einstein, Tolkien, Bohr, Heisenberg (and his cat), Gödel, Kripke (and his worlds)). But we can save the planet Earth of 128 years ago by stating that the information at that time was notably greater than what it was 256 years ago (no Beethovens, nor Hilberts, nor Puccinis, nor Cantors, nor Gausses, nor Rimsky-Korsakovs... Mozarts). For the information available on planet Earth about 256 years ago there was no game in beating the information of 512 years ago, when certain Newton, Gauss, Galileo, Fermat, Copernicus were yet to
born (and when we knew nothing about Königsberg and its bridges). What can we say in defence of the information available on planet Earth 512 years ago? Well, the information available on planet Earth 512 years ago was tremendously greater than the information available on planet Earth about 576 years ago.

Now, stop. Why 576 years ago and not 1024? The reason is quite simple. Because it occurred that around 1439 anno domini a notable German by the name of Johannes Gutenberg invented the mechanical movable type printing, which in turn started the Printing Revolution.\(^1\) Up to then, books were made with ink and quills by scribes working in monastery scriptoria. A day of hard work produced nothing more than few pages. Thanks to Gutenberg’s invention this number drastically increased up to 3600 printed pages per day. Books of “bestselling authors” such as Luther and Erasmus were sold by the hundreds of thousands in their lifetime [Febvre, 1976, Issawi, 1980, Duchesne, 2006].

Now the question is: was all this possible solely because of the newly invented movable printing press? Or was this all due to another “idea” which came about soon after? Let us step forward of about 150 years. The year was 1605, and Johann Carolus published what is recognised as the first newspaper.\(^2\) How was that possible? How did Carolus manage to collect information, write the articles, print all the daily copies, and finally hand out the copies at the same time?

Suppose that Carolus went around Strassburg collecting information from early morning through early-afternoon. Suppose he had two colleagues doing the same. After having collected the information, the three gentlemen went to their “headquarters” to write the articles. Let us say that the writing process lasted a couple of hours, so the articles were done by 5-6pm. Then it was Sebastian, Pyotr, and Ludwig’s turn. Their task was to assemble the type metals to form a printable page; suppose they produced a page every 40 minutes. Once a page was ready to be printed, that was given to whichever of the two-people team was free (Amadeus-Nikolas and Peter-Fryderyk and Francisken-Modest). A team, we say, was able to print up to 200 pages per hour. At 3am, all the printings were done, the newspapers were assembled by the same teams, and by 5am all the copies were ready to be delivered.

If now we search the Oxford English Dictionary for a term to define “a series of actions or steps taken in order to achieve a particular end”, or “a systematic series of mechanised operations that are performed in order to produce something”, we find the noun process. Instead, a query for “an activity that someone is engaged in”,

---

\(^1\)Here, we give credit to Johannes Gutenberg’s improved version of the movable type system of printing for the Printing Revolution, but we recognised the earlier contribution by Bi Sheng.

\(^2\)Relation aller Fürnemmen und gedenckwürdigen Historien.
or for “trade considered in terms of its volume and profitability” returns the same output: business.

Computer scientists, who are typically efficient at combining different information to get new one, define a business process or a business method as a collection of related, structured events and activities that produce a specific service or product for a particular customer or customers.

A business process begins with a mission objective and ends with the achievement of the business objectives (goals). It can often be visualised with a workflow consisting of a sequence of connected steps, where each step follows the precedent and ends just before the subsequent step may begin. In this perspective, tasks can be linked together to create dependencies.

Indeed, as the above scenario shows, a business process involves a series of events and activities. We distinguish between event and activity by stating that the former has no duration (the newspapers ready to be delivered may be seen as an event), while the latter takes time (the composition of type metals or the printing through the wooden press are two distinct activities).

When an activity is rather simple and can be seen as one single unit of work, we call it task. As the whole business process, a task has its own inputs and outputs. To be allowed to start its execution, a task requires that some preconditions are satisfied. For instance, to begin the composition of the type metals together to form a printable page, Sebastian, Pyotr and Ludwig needed as “input” one handed written page by either Carolus or his colleagues. When a task finishes its execution, it changes, somehow, the state of the world by bringing about some effects. For instance, after 40 minutes of work, Sebastian, Pyotr and Ludwig produced a printable page.

In addition to events, a process requires decision points, those being points in time when some crucial decision is made which will affect the future execution of the process. Typically, such decisions are made by people with the necessary level of authorisation. Various levels of authorisation define the role an actor plays within the organisation. For instance, neither Modest nor Amadeus were authorised to decide if an article needed grammatical corrections. That decision was pertaining only to the chief editor, Carolus. Modest and Amadeus could only decide which wooden printing press to use.

Business processes are typically useful for two reasons. First, they provide a good source of information about activities and capabilities of an organisation. Second, such information is used to improve them. Business Process Management (BPM) can be described as a “process optimisation process”. Being a holistic
CHAPTER 1. INTRODUCTION

managerial approach, BPM considers processes as strategic means of an organisation that must be understood, analysed, and improved to furnish always better and more desirable products to clients. These processes are critical to any organisation as they often represent a significant proportion of costs.

For the benefits brought by BPM to be effective, suitable representations of business processes should be given. While an experienced programmer writes thousands of lines of code, a typical user (or process owner) does not want, or has the ability to analyse complicated or convoluted formulas. They instead want simple representations, easy to understand. In this sense, Business Process Modelling technology emerged as a strong paradigm for the modelling, analysis, improvement, and automation of the day-to-day activities of organisations. The field is now a mature research area with a widespread adoption in industry. Business Process Modelling covers a wide variety of methodologies: From graphical modelling languages to ease the understanding of the stakeholders (e.g., YAWL [Russell et al., 2009], EPC [van der Aalst, 1999], BPMN\(^3\)) to fully precise mathematical formalisms (e.g., Petri Nets [van der Aalst, 1998], \(\pi\)-calculus [Milner et al., 1992a,b]) for formal analysis of the properties and automated verification of processes.

All the above mentioned formalisms and representations fall in the family of imperative approaches: They define a process model as a detailed specification of a step-by-step procedure that should be followed during the whole execution. In such a way, they strictly specify how the process will be executed.

If from one side this procedural nature is their strength, it is also their main drawback. In fact, they suffer from some limitations. First, it is sometimes hard to obtain precise information about the order of the actions to be performed from the business requirements. Second, such a paradigm is not suitable to capture flexible business processes, i.e., processes whose internal structure and relationships among the various tasks is dynamic and with a large degree of variations (e.g., triage processes in hospital emergency rooms). Third, their imperative nature yields over-specified and highly-structured processes [van der Aalst and Pesic, 2006] where it is difficult to define relationships among the atoms. For example, it is possible to model a simple statement as “activities A and B should never occur together” only through a detailed strategy to implement it. Finally, they are not flexible enough to efficiently handle runtime exceptions.

The last point is crucial. The first, and foremost, concern of those languages is the process in its whole, and they do not give particular attention to the atoms

\(^3\)http://www.bpmn.org
composing it: The individual tasks. The opposite direction moves the school of modelling processes by declarative specifications [Pesic et al., 2007, Chesani et al., 2008, Governatori et al., 2013b]. Instead of specifying a process step by step, the focus in this approach is on defining relationships among the tasks to be executed to achieve a goal, as well as in understanding the behaviour of such “atoms”.

By shifting the focus from the whole process to its basic building blocks, you gain knowledge regarding which preconditions trigger the activation of a task, as well as what happens once a task completes its execution. Again, it is typical that a task needs some input to begin the execution and “transforms” such inputs into outputs, by performing its internal operation(s). This way of describing processes, in turn, provides knowledge concerning the reasons why to execute a task instead of another and, implicitly, defines the control flow of a process as all the situations that do not violate any of the given relationships. Examples are temporal relationships between tasks (e.g., before, after), co-occurrence/absence, dependency and so on. For a seminal work in this area see [van der Aalst et al., 2009]. Thus, in this paradigm there is a switch from how (procedural) to what (declarative).

This is the most significant difference between the two approaches. Whereas imperative languages look at processes as the starting point, declarative approaches start from a collection of specifications describing the capabilities of an organisation and only then derive the process graph. By doing so, the process itself gains some sort of flexibility, and adaptability. When a task fails because some conditions of the working environment have changed, such a task can be “placed” where its activation preconditions are satisfied (it is no more a “static element” of the process). But we have more. What happens if this is not possible? Since we know what has changed as well as the initial collections of capabilities, it may be possible to substitute such a task with another one, or to call an alternative sub-process. Here, our final aim is to create on the fly a specialised process for particular situations, and as such this research can be seen as (at least the first steps in the direction of) defining processes at run time.

Another important value of this approach is that we can combine business specifications with normative specifications within a single framework. This is a crucial aspect of BPM for two reasons. From one side, the field of Business Process Compliance studies that the business practices are not in breach with the legislation regulating the organisational environment [Governatori and Sadiq, 2008, Sadiq and Governatori, 2015]. Worldwide scandals such as Societe Generale
(France), Enron (USA) and HIH (Australia) forced governments and standard organisations to enact more restrictive regulatory mandates leading enterprises to massive investments in the market of compliance related software and services (over $30 billion in 2008 [Hagerty et al., 2008]). Scholars have studied automated methods to establish if a business process is compliant or not with the norms ruling the environment where the organisation acts in [Governatori et al., 2009a, Governatori and Rotolo, 2010b] and BPC deals with the problem of developing the above mentioned methods.

On the other side, it is meaningless to specify a business by only considering normative aspects while ignoring the final objectives of the organisation. This may lead to obtain processes fully compliant with the normative system but meaningless from the “business perspective”. For instance, let us consider managing the business process of a library where a (plausible internal) policy states that each book must be returned within a month. A (non-plausible) solution would be not to lend books anymore (in clear contrast with the final aim of being a library).

Compliance requirements are usually formulated as a set of rules that can be checked during or after the execution of the business process, called compliance by detection. If a non-compliant behaviour is detected, the business process needs to be redesigned. Alternatively, the rules can be already taken into account while modelling the business process to result in a business process that is compliant by design (being this the direction of my Ph.D. research). This technique, which goes under the name of compliance by design, has the advantage that a subsequent verification of compliance is not required.

Before entering in the detail of compliance by design, let us analyse all the three criteria that may be adopted when dealing with compliance issues. In general, a compliance regimen include three distinct but interrelated perspectives, those being detective, corrective, and preventative measures.

Corrective measures are intended to limit the extent of any consequence caused by a non-compliant situation. That situation can arise in front of the introduction of a new norm impacting upon the business, to the organisation coming under surveillance and scrutiny by a control authority or to an enforceable undertaking. Corrective measures, undertaken in a proactive manner, position the organisation favourably with regulators or other control authorities.

Detective measures are intended to identify an “after-the-fact” non-compliant situation. There are two main approaches: (i) Retrospective reporting through manual audits by consultants or through IT forensics and Business Intelligence
tools, (ii) *Automated detections* generating audit reports against hard-coded checks performed on the requisite system. The proposed solutions hook into a variety of enterprise system components (e.g., SAP HR, LDAP Directory, Groupware, etc.). Unlike the first approach, automated detection reduces the assessment time and consequently also the time of non-compliance remediation/mitigation.

These two approaches suffer from the lack of *sustainability*, caused by the extreme interest of companies to continuously improve the quality of service, and for changing legislatures and compliance requirements. Indeed, even with automated detection means, the hard coded check repositories can quickly grow to a very large scale making it extremely difficult to evolve and maintain.

Moreover, if we consider the case of a company that wants to implement new services or in the case where flexible processes should be defined, the information at hand is not contextualised in a fully formalised process but in the form of *repository of capabilities*: Actions and relations among them, preconditions and effects of a particular operation, activities, or fragments of processes (sub-processes).

Let us consider the scenario of an hospital. Along with procedures that can be managed with prefixed patterns (e.g., storage room, routine medical examinations), there are situations where the undertaken procedure should be dynamically adapted under the circumstances at hand (e.g., triage procedures). Roughly speaking, the generation of the process must be *ad hoc* and *on the fly*.

For these reasons in [Sadiq et al., 2007, Governatori and Sadiq, 2008, Sadiq and Governatori, 2015], the authors suggested that a suitable approach for achieving compliance should have a *preventative focus*. A solution to this issue is to embed the concept of compliance into the business practice. Follows the central research statement.

*Starting from a description that represents capabilities and resources of a company, the regulations governing the environment the enterprise acts in, and the business objectives the company wants to achieve, the present Ph.D. dissertation will face the problem of defining formal methods to generate an entire process compliant both with norms and goals (compliance by design).*

The initial point is a compliant situation, i.e., with no breach in the regulative system and with all the means necessary to reach all the business objectives. Thus, after the building process ends up, there is no need of checking compliance, since the business process will be compliant by design.
This doctorate dissertation is presented as follows. We introduced in the first chapter the motivational background, thus we shall explain in the next chapter the methodology adopted, along with some logical background on Defeasible Logic and the notation we shall adopt to represent graphs. If in the introduction we have justified why adopting a logical language to describe declarative specification is a good choice, in Chapter 2 we shall make a step further by explaining what “binds” logic to graphs, derivations to (process) traces.

Before passing to the technical part of the thesis, we shall propose a thorough motivational investigation, being that Chapter 3, of why we chose to develop our modal defeasible apparatus. There, we shall present our ideas underlying choices we made, describe every and each modality introduced and relate them with one another. Finally, we shall compare our framework against other approaches in the relevant literature of the field.

Then come Chapters 4 and 5. The former chapter will formalise our logic, while in the latter chapter we shall propose the algorithms to compute the extension sets, that is what we can prove and what we can disprove given a specific situation (set of facts). Lemmas and theorems regarding the correctness of such algorithms are proposed in the final part of Chapter 5, but to ease the reading the corresponding demonstrations are postponed to Appendix A.

In Chapter 6, we shall advance the final results of this Ph.D. dissertation, those being on how to transform the declarative specifications into a process graph. The algorithms proposed in that chapter perform two types of operations.
The first type of operations is how to create a business process by taking into account all the relevant tasks, conditions, business objectives and obligations in force. The second type of operations are meant to simplify the process graph by creating parallel and choice patterns and by identifying subsets of tasks which occur multiple times together.

Chapter 4 and Chapter 6 are ended by Section 4.4 and Section 6.3 which propose two running examples.

We chose to discuss relevant literature and future lines of research with specific sections in the corresponding chapters (namely, Section 4.5 for the logic, and Section 6.5 for the process synthesis algorithms).

Chapter 7 will conclude the dissertation by summarising the work done; in there, we shall propose a reflection about our contribution and outline future directions of research.

List of relevant publications

Part of the work presented in this doctoral dissertation has been published in various international conferences. At the beginning of each chapter, the reader will find the reference of the publications related with the content of the chapter itself. Below is a complete list of the published works.


CHAPTER TWO

CONCERNING METHODOLOGY

The key characteristic of the compliance by design approach is its ability to capture compliance requirements through a generic modelling framework which subsequently facilitates the propagation of these requirements into business process models. In this approach the greatest challenges lie in aligning control objectives that stem from regulations and legislation with business objectives devised for improved business performance.

To succeed in our endeavour, we need to move in two directions. The initial challenge consists of understanding how to represent processes and organisational capabilities in a formal way. By following the trend of declarative specifications, the capabilities discussed in the previous chapter can be encoded in a fully declarative manner. Through this perspective, a logical approach fully captures the idea of control flow definition in terms of satisfiability over a set of formalised constraints.

The second challenge is how to “transform” such a logical description into a process graph. In fact, the result of the first stage will be logical formulas describing which conditions hold, which tasks are executable, as well as which norms the organisation needs to comply with. These will be presented in the form of logical derivations, where each derivation can be seen as the simulation of an execution trace. Thus, we need to “put together” such information in order to obtain a structured process, i.e., a process where the traces are organised with parallel and choice patterns. We shall refer to the typical structure and notation of Business Process Model Notation (BPMN), which will be made clear in the rest of the chapter.

In regards to the logic aspects, the efforts spent by the scholars in this research area were mainly focused on temporal logics [van der Aalst and Pesic, 2006] and
on languages based on temporal features [Chesani et al., 2008]. In those works, the authors link together logic formalisms and graph theory-like representations that allow the graphical representation of all the possible traces (instances) of a process. The motivation behind their choice lies on the nature of these two formalisms.

In fact, given a generic deductive logic system and an initial set of facts (indisputable statements, which can be seen as inputs for a particular setting), the derivation (or formal proof) of a statement is the final phase of a finite sequence of sentences/steps each of which is a fact or follows from the preceding sentences in the sequence by the application of a rule. A typical rule consists of a set of preconditions, or antecedents, and some conclusions or postconditions. Whenever such preconditions are satisfied, the rule “fires” and produces its conclusion; absent the preconditions the action cannot be taken, and if it is taken the postconditions hold. As such, a derivation has a strong, semantical correspondence with a trace of a process, and thus we can establish a bijection between a process and a logic theory. This is in line with the definition of (business) process we gave in the previous chapter. A task is the result of the successful execution of previous tasks (preconditions) and, in turn, may take part in the activation of one or more other tasks (the various steps in a derivation).

To clarify, let us formalise part of the printing process of Carolus, specifically the part when the type metals composition of a page has been made and such a page needs to be printed by one free team among Amadeus-Nikolas, Peter-Fryderyk and Francisken-Modest. Suppose that the theory formalising the above description consists of the following three rules

\[ r_1 : \text{If } \text{ready}(X), \text{Amadeus - Nikolas\_free} \text{ then printed}(X) \]
\[ r_2 : \text{If } \text{ready}(X), \text{Peter - Fryderyk\_free} \text{ then printed}(X) \]
\[ r_3 : \text{If } \text{ready}(X), \text{Francishen - Modest\_free} \text{ then printed}(X). \]

Each single rule states that if page \( X \) is ready and the team is not occupied in printing another page, then page \( X \) will actually be printed. It is straightforward to model such a schema of rules with the representation proposed in Figure 2.1. Here, we represent tasks as rectangles, while the X-OR gateway depicts the three possible ways a page can be printed (if it is printed by a team, it cannot be printed by another). Moreover, we assume that the execution of task \( \text{Compose Page } X \) gives the output \( \text{ready}(X) \), meaning that page \( X \) is ready to be printed. To execute task \( \text{PrintPageX} \), we have to satisfy two conditions: (i) Task \( \text{ComposePageX} \) must have finished its execution, (ii) One of the three teams must be available. When both conditions are fulfilled, task \( \text{PrintPageX} \) may start its execution, which will
produce $\text{printed}(X)$.

Another distinguishing characteristic is that a logical approach allows us to study and verify the formal properties of the proposed framework (e.g., consistency and correctness), while the inferential mechanism determines the conclusions of a particular situation.

For the scope of our research, the main focus is in studying the semantical relationships among the elements of a process (logical relationships between preconditions and tasks, tasks and their effects, triggering of obligations) rather than capturing temporal or precedence constraints among tasks. Thus, we shift our attention on Defeasible Logic [Antoniou et al., 2001]. We shall propose an extension that enriches and combines the deontic Defeasible Logic of violations [Governatori, 2005] for modelling contracts and which was used for regulatory compliance of processes by Governatori et al. [2006], Sadiq et al. [2007], and the defeasible BIO (Belief-Intention-Obligation) logic for modelling agents [Governatori and Rotolo, 2008b].

The rest of the present chapter will proceed as follows. The next section will introduce the first notions of Defeasible Logic, the logical formalism we chose to represent the declarative aspects regarding the organisation capabilities. It is ended by a brief introduction of the modal variant we use to design our own logical formalisation. In Section 2.2, we shall motivate how to pass from the logical representation to the graph theory-like one. In there, we shall introduce the BMPN-like notation adopted in order to represent our process graphs.

2.1. Of the defeasibility of knowledge: The case of Defeasible Logic

The non-monotonic formalism of Defeasible Logic was introduced in a seminal work by Nute [1987]. A defeasible theory consists of five different kinds of
knowledge: Facts, strict rules, defeasible rules, defeaters, and a superiority relation. Examples of facts and rules below are standard in the literature of the field.

Set PROP defines propositional atoms, while Lab is a set of arbitrary labels. The set Lit = PROP∪{¬p|p ∈ PROP} denotes the set of literals. The complement of a literal q is denoted by ¬q; if q is a positive literal p, then ¬q is ¬p, and if q is a negative literal ¬p then ¬q is p.

A defeasible theory D is a structure (F, R, >), where

1. F ⊆ Lit denotes simple pieces of information that are considered always to be true. For example, a fact is that “Sylvester is a cat”, formally cat(Sylvester);


3. >⊆ R × R is a binary, antisymmetric relation.

A theory is finite if the set of facts and rules are so.

A rule is an expression r : A(r) ⇐ C(r) and consists of: (i) A unique name r ∈ Lab, (ii) The antecedent A(r) which is a finite subset of Lit, (iii) An arrow ⇐∈ {→, ⇒, ¬⇒} denoting, respectively, a strict rule, a defeasible rule and a defeater, and (iv) Its consequent (or head) C(r) ∈ Lit, which is a single literal. A strict rule is a rule in which whenever the premises are indisputable (e.g., facts), then so is the conclusion. For instance,

\[\text{cat}(X) \rightarrow \text{mammal}(X)\]

means that “every cat is a mammal”. On the other hand, a defeasible rule is a rule that can be defeated by contrary evidence; for example, “cats typically eat birds”:

\[\text{cat}(X) \Rightarrow \text{eat}_\text{birds}(X)\]

The underlying idea is that if we know that something is a cat, then we may conclude that it eats birds, unless there is evidence proving otherwise. Defeaters are rules that cannot be used to draw any conclusion. Their only use is to prevent some conclusions, i.e., to defeat defeasible rules by producing evidence to the contrary. An example is “if a cat has just fed itself, then it might not eat birds”:

\[\text{just} \_ \text{fed}(X) \Rightarrow \neg\text{eat}_\text{birds}(X)\]

The superiority relation > among rules is used to define where one rule may
override the (opposite) conclusion of another one, e.g., given the defeasible rules

\[
\begin{align*}
 r &: \text{cat}(X) \Rightarrow \text{eat}_\text{birds}(X) \\
 r' &: \text{domestic\_cat}(X) \Rightarrow \neg\text{eat}_\text{birds}(X)
\end{align*}
\]

which would contradict one another if Sylvester is both a cat and a domestic cat, they do not in fact contradict if we state that \( r' \) wins against \( r \), leading Sylvester not to eat birds.

Like in [Antoniou et al., 2001], we consider only a propositional version of this logic, and we do not take into account function symbols. Every expression with variables represents the finite set of its variable-free instances.

The infix notation \( r > s \) means that \( (r, s) \in \triangleright \). The set of strict rules in \( R \) is denoted by \( R_s \), and the set of strict and defeasible rules by \( R_{sd} \). \( R[q] \) is the rule set in \( R \) with head \( q \). A conclusion of \( D \) is a tagged literal and can have one of the following forms:

- \(+\Delta q\), which means that \( q \) is definitely provable in \( D \), i.e., there is a definite proof for \( q \), that is a proof using facts, and strict rules only;
- \(\neg\Delta q\), which means that \( q \) is definitely not provable in \( D \) (i.e., a definite proof for \( q \) does not exist);
- \(+\partial q\), which means that \( q \) is defeasibly provable in \( D \);
- \(\neg\partial q\), which means that \( q \) is not defeasibly provable, or refuted in \( D \).

Given a defeasible theory \( D \), a proof \( P \) of length \( n \) in \( D \) is a finite sequence \( P(1), \ldots, P(n) \) of tagged formulas of the type \(+\Delta q\), \(\neg\Delta q\), \(+\partial q\) and \(\neg\partial q\), where the proof conditions defined in the rest of this section hold. \( P(1..n) \) denotes the first \( n \) steps of proof \( P \).

Given \( \# \in \{\Delta, \partial\} \) and a proof \( P \) in \( D \), a literal \( q \) is \#-provable at line \( n \) in \( D \) if there is a line \( P(m), n \leq m, \) of \( P \) such that \( P(m) = +\#q \). A literal \( q \) is \#-refuted at line \( n \) in \( D \) if there is a line \( P(m), n \leq m, \) of \( P \) such that \( P(m) = -\#q \). When clear from the context, we simply use provable and refuted.

Positive and negative proof tags are related by the principle of strong negation. This is closely related to the function that simplifies a formula by moving all negations to an inner most position in the resulting formula, and replaces the positive tags with the respective negative tags, and the other way around [Antoniou et al., 2000].

The definition of \( \Delta \) describes just forward chaining of strict rules.

\[ +\Delta: \text{If } P(n + 1) = +\Delta q \text{ then} \]

1. \( q \in F \) or
2. \( \exists r \in R_s[q] \forall a \in A(r) : +\Delta a \in P(1..n) \).
Literal $q$ is definitely provable if either (1) is a fact, or (2) there is a strict rule for $q$, whose antecedents have all been definitely proved.

$-\Delta$: If $P(n+1) = -\Delta q$ then

(1) $q \notin F$ and
(2) $\forall r \in R_i[q] \exists a \in A(r) : -\Delta a \in P(1..n)$.

Literal $q$ cannot be definitely proven ($-\Delta q$) if (1) is not a fact and (2) every strict rule for $q$ has at least one definitely refuted antecedent.

The following statements define notions of being applicable and discarded.

In the proof condition for $\pm \partial$, a rule $r \in R_{sd}[q]$ is (i) Applicable iff $\forall a \in A(r)$, $+\partial a \in P(1..n)$; (ii) Discarded iff $\exists a \in A(r)$ such that $-\partial a \in P(1..n)$.

We now introduce proof tags for the defeasible provability.

$+\partial$: If $P(n+1) = +\partial q$ then

(1) $+\Delta q \in P(1..n)$ or
(2) (2.1) $-\Delta \sim q \in P(1..n)$ and
(2.2) $\exists r \in R_{sd}[q]$ s.t. $r$ is applicable, and
(2.3) $\forall s \in R[\sim q]$. either $s$ is discarded, or
(2.3.1) $\exists t \in R[q]$ s.t. $t$ is applicable and $t > s$.

$-\partial$: If $P(n+1) = -\partial q$ then

(1) $-\Delta X q \in P(1..n)$ and either
(2) (2.1) $+\Delta \sim q \in P(1..n)$ or
(2.2) $\forall r \in R_{sd}[q]$. either $r$ is discarded, or
(2.3) $\exists s \in R[\sim q]$ s.t. $s$ is applicable, and
(2.3.1) $\forall t \in R[q]$. either $t$ is discarded, or $t \not> s$.

Literal $q$ is defeasibly provable if (1) $q$ is already definitely provable, or (2) we argue using the defeasible part of the theory. For (2), $\sim q$ is not definitely provable (2.1), and there exists an applicable strict or defeasible rule for $q$ (2.2).

Every attack $s$ is either discarded (2.3), or defeated by a stronger rule $t$ (2.3.1).

As usual, given a proof tag $\#$, a literal $p$ and a theory $D$, we use $D \vdash \pm \# p$ to denote that there is a proof $P$ in $D$ where for some line $i$, $P(i) = \pm \# p$. Alternatively, we say that $\pm \# p$ holds in $D$, or simply $\pm \# p$ holds when the theory is clear from the context.

The set of positive and negative conclusions is called extension. Given a defeasible theory $D$, the (defeasible) extension of $D$ is defined as $E(D) = (+\Delta, -\Delta, +\partial, -\partial)$, where $\pm = \{l : l$ appears in $D$ and $D \vdash \pm \# l\}$, $\# \in \{\Delta, \partial\}$.

The inference mechanism of Defeasible Logic does not allow to derive inconsistencies unless the monotonic part of the logic is inconsistent. A defeasible theory $D$ is inconsistent iff there exists a literal $p$ such that $(D \vdash \pm \# p$ and
Defeasible Logic seems to be the proper tool due to some of its distinctive features:

- Defeasible Logic is based on a logic programming-like language and it is a simple but flexible non-monotonic formalism capable of dealing with many different intuitions of non-monotonic reasoning.
- Defeasible Logic has a linear complexity [Maher, 2001] and several studies show that the complexity does not increase in most cases when Defeasible Logic is enriched with modal operators [Governatori and Rotolo, 2008b,a] or temporal operators [Governatori and Rotolo, 2010c].
- Defeasible Logic permits declarative specifications which are executable; unlike most other logic approaches, Defeasible Logic was designed to be easily implementable right from the beginning (above all implementations, we mention Deimos [Rock, 2000], Delores [Maher et al., 2001] and SPINdle [Lam and Governatori, 2009]).

Governatori and Rotolo [2006] propose a modal variant to describe agents. They introduce modalities such as beliefs, intentions and obligations. The main contribution of that work is to describe contrary-to-duty obligations through reparative chains of obligations. The formalisation of reparative chains is very efficient in understanding which obligation must be activated in order to recover from the violation of a previous obligation.

Propositional atoms in our theory stand for tasks and conditions. The logical negation of an atom denotes a task that can never execute or a non valid state (in the case of conditions). Atoms and their negations are the literals of the theory. A modal literal represents a task that must execute (a condition that must be reached) in case of obligations, or the desire of the organisation to execute a task (or to achieve a condition) in case of objectives.

An inference engine must be devised to compute new factual knowledge, objectives and obligations. To this end we use three types of rules: belief rules, obligation rules and outcome rules.

Belief rules are used to relate the factual knowledge of an enterprise, composed by

- the set of actions an organisation can do;
- the preconditions under which tasks can be executed;
- the effects derived by the execution of these tasks (also called postconditions).

Specifically, belief rules describe the logical relationship between preconditions
and tasks, tasks and their effects, relationships between tasks, relationships between states. *Obligation rules* determine when and which obligations are in force, while *outcome rules* establish the objectives of an organisation depending on the particular context. A literal can be modalised through the application of obligation or outcome rules (for example, the execution of task $t$ leads to the obligation of condition $c$, in notation $Oc$). A theory corresponds to a normative system, i.e., a set of norms, where every norm is modelled by rules. The *superiority relation* is extended to detect and solve conflicts between different types of rules.

### 2.2. Concerning process graphs representation

The core of this dissertation is the investigation of how to generate business process models starting from the declarative specifications of the environment, regulations, and the capability of an organisation. A possible solution, called *backward graph approach*, is to start from a derivation and to use patterns of rules present in a theory to create graphs corresponding to the compliant executions of the theory.

In *graph theory*, a *graph* is a representation of a set of objects where some pairs of objects are connected by *links*. The interconnected objects are represented by mathematical abstractions called *vertices*, or *nodes*, and the links that connect some pairs of vertices are called *edges* (*arcs*). Typically, a graph is depicted in diagrammatic form as a set of dots for the vertices, joined by lines or curves for the edges. The edges may be directed or undirected. We shall focus only on directed graphs. In a directed graph, edges have a direction associated with them. An edge $e = (x, y)$ is considered to be directed from $x$ to $y$; $y$ is called the *head* while $x$ is called the *tail* of the arc. If a path made up of one or more successive arcs leads from $x$ to $y$, then $y$ is said to be a *successor* of $x$, and $x$ is said to be a *predecessor* of $y$. The arc $(y, x)$ is called the arc $(x, y)$ inverted.

Business processes consist of separate activities, each of which is an action representing a semantical unit at some level. In addition, an activity can be thought of as a function that modifies the state of the process, making some conditions true, others false, and allowing some tasks to start their execution. Business processes are modelled as graphs with individual activities as nodes. The edges on the graph represent the potential flow of control from one activity to another.

The modal defeasible theory we start with is rich of information: There are lit-
erals describing tasks and conditions, rules describing the activations of tasks and their effects, and rules triggering norms and business objectives. Moreover, the superiority relation states conditions under which a rule is activated (preferred) instead of another as well as patterns on the rules which allow us to identify parallel and choice structures.

We want to use all this information to build the business process. The idea is to start from a theory describing the capabilities of the business. From such a theory we extract the set of all proved literals, i.e., we compute the positive extension of the theory. Then, we consider the set of reachable objectives and we create a node for each objective. For each of these objectives, we find out every rule which proves it whose premises are all in the positive extension of the theory, and we store them. For every antecedent that is also a literal representing a task, we create a new node (if not already in the graph) and we link that node with a directed edge from it to the corresponding objective node. For every new node, we iterate the process (navigating backwards the derivation). The procedure terminates when we exhaust the literals in the positive extension.

Some observations that underpin the above technique follow. The precedence constraint represented by the edge between two nodes when we link them during the construction of the graph is a very relaxed relation of response [van der Aalst and Pesic, 2006]; for example if \( a \) is linked to \( b \), the intended meaning is not “\( b \) has to execute immediately after \( a \)”, but “\( a \) is eventually followed by the execution of \( b \)”. Suppose we have a graph where an edge between \( a \) and \( b \) exists as well as an edge between \( c \) and \( d \). Two valid execution sequences are \( a, b, c, d \) and \( a, c, b, d \). In fact at this level of analysis, we are not interested in strict constraints on execution time or the order in which the tasks execute.

We conclude this section by presenting the simplified, and slightly modified version of the BPMN notation we shall adopt.

### 2.2.1. Process graph notation

Business processes includes events and activities. We denote both under the common name of tasks. Tasks are logically related as in graphs. The basic relation among such elements is that of sequence.

Here, task \( A \) is followed by task \( B \), which in turn is followed by task \( C \). Each task is represented by a rectangle, the connection between them is a directed arc. We may interpret the logical dependency between \( A \) and \( B \) by saying that \( B \) may begin its execution only when task \( A \) has finished. Sometimes it is meaningful to attach labels to nodes or edges, as is the case for the arc between \( B \) and \( C \). By
labelling such an edge we impose that $C$ may begin its execution only after $B$ has finished but with the additional constraint that condition $c$ must be the case.

Activities may not necessarily be sequential with one another. For instance, in the “printing” example of the previous chapter, the activity of gathering information by Carolus should happen at the same time as his two colleagues. Moreover, two activities may have the same outcome but only one of them should happen (an activity may exclude another). For example, when a page of an article is ready, then only one among Sebastian, Pyotr, or Ludwig has the task of assembling the type metals to form the printable page.

We say that two or more tasks concur whenever they are not interdependent. This introduces the notion of gateway. We introduce two types of gateways: split gateways and join gateways. A split gateway represents a time in the process where the process flow branches off, while a join gateway corresponds to a point when the process flow merges. We represent such gateways with diamonds.

There are three distinct types of split and join gateways. First, we have parallel (AND) patterns. They represent situations when two or more activities do not have any order dependency and may be executed concurrently (in any order). The notation for AND split and AND join gateways are, respectively, $\text{And-S}$ and $\text{And-J}$.

Here, we have two admissible execution flows, those being $ABC\, D$ and $ACBD$. In both situations, for task $D$ to start its execution, both task $B$ and $C$ must end.
2.2. CONCERNING PROCESS GRAPHS REPRESENTATION

Figure 2.4.: XOR split/join gateways

their execution.

Second, exclusive (decision) (XOR) patterns. Through these patterns, we model mutual exclusive courses of action. The notation for XOR split and XOR join gateways are, respectively, XOR-S and XOR-J.

The two admissible execution flows are \(ABD\) and \(ACD\); notice that there is no trace with both tasks \(B\) and \(C\) occurring at the same time. The execution of one of them excludes the execution of the other.

Finally, we have simple (inclusive) choice (OR) patterns. We may consider them an entity in between the parallel pattern and the exclusive decision pattern. When entering an OR pattern, we may take one or more branches, i.e., at least one of the branches must be executed, without having the constraint that each branch must be executed or that only one branch must be executed. The notation for OR split and OR join gateways are, respectively, OR-S and OR-J.

Figure 2.5.: OR split/join gateways

Here we have four admissible traces: \(ABD\), \(ACD\), \(ABCD\), and \(ACBD\).
TOWARDS A LOGICAL FRAMEWORK TO MODEL DECLARATIVE SPECIFICATIONS

Bilbo’s eleventhy-first long-expected birthday party is almost over and Frodo has just received as legacy The One Ring. What to do with such a powerful and fearful object? Being that his Baggins side is still hidden inside him, Frodo thinks that The Ring would be safer in Gandalf’s hands. If this is not possible, then he can undertake the quest to destroy it in Mount Doom. Gandalf being even wiser than powerful refuses to be entrusted with The Ring; thus Frodo leaves Bag End with his buddy, Sam.

Once in Rivendell, House of Elrond, Frodo and the Fellowship of the Ring must decide which is the best path to get to Mordor. The first peril which they come upon are the frosty Misty Mountains. Frodo’s first choice would be for the Fellowship to climb them. If not climb them, pass South of them, leaving the last and least favourite alternative of passing through Dwarrowdelf.

The Fellowship tries to climb the Misty Mountains but fails due to some very unlucky weather conditions, the path being closed until the coming of Summer. Then, Gandalf tells Frodo that he may not go South since Saruman would imprison him and take The Ring for himself. It seems that, in the end, the Fellowship will have to sneak through Moria without being noticed by orcs and trolls.

The previous example reveals a very (un)common situation: A system operates in an environment to reach some objectives. During this process, the system must interact with some constraints imposed by the environment. For instance,

---

1I personally apologise for the rough synopsis of what I believe to be one of the most influential books of my life [Tolkien, 1954].
very bad weather may prevent the system from climbing the mountains. But not only that, the system may not be the only actor in play. Other entities can change, either directly or indirectly, the conditions of the environment; some of them may even have the power to state constraints or policies the system must obey while accomplishing its purposes.

The system may have its own desires and intentions, and the combination of such mental attitudes with the factuality of the world defines its deliberative process, i.e., which objectives the system will commit to, which ones will be estimated as unfeasible. Even once the system will have understood which objectives are attainable, the system may give up some of them to comply with the norms.

At last, since many contexts prevent the system from achieving all its objectives, the system must then understand which objectives are mutually compatible with each other and choose which objectives to attain the least of in given situations by ranking them in a preference ordering.

The problem we shall address in this, and the next, chapter is how to formally describe a system operating in an environment, with some objectives to achieve, and possibly not violating the norms governing the domain in which the system operates.

To model such systems, we have to specify three types of information: (i) The environment where the system is embedded, i.e., how the system perceives the world, (ii) The norms regulating the application domain, and (iii) The system’s internal constraints and objectives.

This chapter is based on [Governatori et al., 2013b].

Outline

The present chapter is structured as follows. Section 3.1 gives a general picture of how scholars of the field have modelled cognitive agent systems, and shows limits of the current agent literature. It justifies our choices against other approaches to choose the concept of outcome as building block of our agent’s motivational attitudes, which is simply something that an agent expects to achieve or that can possibly occur. This is motivated by our conviction that all agent’s outcomes should be derived by “mixing up motivational attitudes, factuality of the environment and norms altogether”.

Section 3.2 proposes the underlying ideas of our framework, where we shall explain in detail, without formalisation, the phases an agent should pass through and what she should consider when she deliberates about her outcomes, those be-
3.1. Introducing Outcomes: A motivational detour

The core component of a rational system is its capability to make rational choices of actions in order to satisfy its design objectives. A successful abstraction to represent a system operating in an environment where the system itself must exhibit some kind of autonomy is that of BDI (Belief, Desire, Intention) architecture [Rao and Georgeff, 1991] inspired by the work of Bratman [1987] on cognitive agents. Desires and intentions model the agent’s mental attitudes and are meant to capture the objectives, whereas beliefs describe the environment. More precisely, the notions of belief, desire, and intention represent respectively the informational, motivational, and deliberative states of an agent [Wooldridge and Jennings, 1995].

Over the years, several frameworks, either providing extensions of BDI or inspired by it, were given for extending models for cognitive agents to also cover normative aspects (see, among others, [Broersen et al., 2002, Thomason, 2000, Governatori and Rotolo, 2008b]). This is a way of developing normative agent systems, where norms are meant to ensure global properties for them [Andrighetto et al., 2013]. In these extensions the agents’ behaviour is determined by the interplay of the cognitive component and the normative one (such as obligations). In this way, they represent how much an agent is willing to invest to reach an outcome based on the states of the world (beliefs) and norms. Indeed, beliefs and norms are of the utmost importance in the decision process of the agent. If she does not take beliefs into account, then she will not be able to plan what she wants to achieve (her planning process would be a mere wishful thinking). On the other hand, if the agent does not respect the norms governing the environment she acts in, then she may incur sanctions from other agents [Bratman, 1987].

The BDI approach is based on the following assumptions about the motivational and deliberative components. The agent defines *a priori* her desires and intentions, and only after this is done the system verifies their mutual consistency by using additional axioms. First, such entities are not inter-related with one another since “the notion of intention […] has equal status with the notions of belief and desire, and cannot be reduced to these concepts” [Rao and Georgeff, 1991]. Second, the agent may consequently have intentions which are
contradictory with her beliefs and this may be verified only \textit{a posteriori}.

Traditional formal treatments of the BDI model, such as [Cohen and Levesque, 1990, Rao and Georgeff, 1991, Wooldridge, 2000], attempt to capture in a multi-modal logic framework the static and dynamic properties of beliefs, desires and intentions, and the relationships that are required to hold between these elements. Beliefs are typically axiomatised in such a way that are effectively distinct from knowledge, and are consistent, closed under consequence and introspection, whereas intentions, whether seen as primary or derived, are usually subject to weaker requirements such as consistency. In those frameworks, intentions are required to be logically possible and actually achievable in order to be assumed, and should be dropped whenever new beliefs “collide” with previously adopted intentions.

Hence, one of the main conceptual deficiencies of the BDI paradigm (and generally of almost all classical approach to model rational agents) is that the deliberation process is bound to mental attitudes which are independent (meaning that none of them is fully definable in terms of the others) and fixed \textit{a priori}.

Approaches like BOID (Belief-Obligation-Intention-Desire) architecture [Broersen et al., 2001] and Governatori and Rotolo [2008b]’s system improved previous frameworks, for example, by structurally solving conflicts between beliefs and intentions (the former being always stronger than any conflicting intentions), while mental attitudes and obligations are just meant to define which kind of agent (social, realistic, selfish, etc.) are admissible.

To begin with, unlike the BDI perspective, we aim at proposing a fresh conceptual and logical analysis of the motivational and deliberative components within a unified perspective.

\textbf{Requirement 1} (A unified framework for agents’ motivational and deliberative components). \textit{Goals, desires, and intentions are different facets of the same phenomenon (all of them being goal-like attitudes).}

\textit{This reduction into a unified perspective is done by resorting to the basic notion of outcome, which is simply something (typically, a state of affairs) that an agent expects to achieve or that possibly can occur.}

Even when considering the vast literature on goals of the last ten years, most of the authors studied the content of a goal (e.g., \textit{achievement} or \textit{maintenance} goals) and conditions under which a goal has to be either pursued, or dropped. This kind of (\textit{a posteriori}) analysis may be compared as orthogonal to ours, since we want to develop a framework computing the agent’s mental attitudes, combining her beliefs and the norms with her desires.
As we shall argue, an advantage of the proposed analysis is that it allows agents to compute different degrees of motivational attitudes, as well as degrees of commitment that take into account other factors, such as beliefs and norms.

**Requirement 2** (Agents’ motivations emerge from preference orderings among outcomes). *The motivational and deliberative components of agents are generated from preference orderings among outcomes.*

As done in other research areas such as rational choice theory, we move with the idea that agents have preferences and choose according to them. Preferences involve outcomes and are explicitly represented in the syntax of the language for reasoning about agents, thus following the logical paradigm initially proposed in [Brewka et al., 2004, Governatori and Rotolo, 2006].

The combination of mental attitudes with the factuality of the world defines the agent’s deliberative process, i.e., the objectives she decides to pursue. If required, the agent may give up some of them to comply with the norms. Indeed, many contexts may prevent the agent from achieving all her objectives; the agent must then understand which objectives are mutually compatible with each other and choose which ones to attain the least of in given situations by ranking them in a preference ordering.

We can distinguish three phases an agent must pass through to bring about certain states of affairs: understanding the environment she acts in, deploying such information to deliberate which objectives to pursue, and how to act in order to reach them.

In the first phase, the agent gives a formal declarative description of the environment (in our case, a rule-based formalism). Rules allow the agent to represent relationships between pre-conditions and actions, actions and their effects (post-conditions), relationships among actions, which conditions trigger new obligations to come in force, and in which contexts the agent is allowed to pursue new objectives. In the second stage, the agent combines the formal description with an input describing a particular state of affairs of the environment, and determines which norms are actually in force, which objectives she decides to commit to and to what degree. The agent’s decision is based on logical derivations. Since the agent’s knowledge is represented by rules, during the last phase, the agent exploits information from derivation to select the activities to carry out in order to achieve the objectives. Here, it is relevant to notice that a derivation can be understood as a virtual simulation of the various activities involved.

While different schemas for generating and filtering agents’ outcomes are
possible, accordingly to the three phases described above we shall restrict ourself to schemas where we adopt the following principles:

- When an agent faces alternative outcomes in a given context, these outcomes are ranked in preference orderings;
- Mental attitudes are obtained from a single type of rule (outcome rule) whose consequents express the above mentioned preference orderings among outcomes;
- Beliefs prevail over conflicting motivational attitudes, thus avoiding various cases of wishful thinking [Thomason, 2000, Broersen et al., 2002];
- Norms and obligations are used to filter social motivational states (social intentions) and compliant agents [Broersen et al., 2002, Governatori and Rotolo, 2008b];
- Goal-like attitudes can also be derived via conversion using other mental states, such as beliefs (e.g., believing that Madrid is in Spain may imply that the goal to go to Madrid implies the goal to go to Spain) [Governatori and Rotolo, 2008b].

Our effort is finally motivated by computational concerns. The logic for agents’ desires, goals, and intentions is expected to be computationally efficient. In particular, we shall prove that computing agents’ motivational and deliberative components in our unified framework has linear complexity.

### 3.2. The intuition underneath the framework

When a cognitive agent deliberates about what her outcomes are in a particular situation, she selects a set of preferred outcomes among a larger set, where each specific outcome has various alternatives. It is natural to rank such alternatives in a preference ordering, from the most preferred choice to the one she deems least acceptable.

Let us consider again the Middle-Earth setting and focus here on the three alternatives Frodo and the Fellowship settle to pass over the Misty Mountains. We saw that: Frodo’s first choice being to climb them, his second one being to pass South through the Gap of Rohan, while as last option Frodo is willing to consider passing under them through Moria.

To represent the scenario above, we need to capture (1) the preferences about his alternatives, and (2) his beliefs about the world. For (1), we build a sequence of alternatives $A_1, \ldots, A_n$ that are preferred when the previous choices are no longer feasible. Normally, each set of alternatives is the result of a specific context
3.2. THE INTUITION UNDERNEATH THE FRAMEWORK

C stating under which conditions (premises) such a sequence of alternatives $A_1, \ldots, A_n$ can be taken into consideration.

Accordingly, we can represent Frodo’s alternatives with the graphic notation

\[ \text{If Ring\_bearer and destroy\_Ring then climb\_MistyMts, Gap\_Rohan, pass\_Moria.} \]

This intuition resembles the notion of obligation chains presented in [Governatori and Rotolo, 2006]. In there, a norm is represented by an obligation rule of the type

\[ r_0 : \text{drive\_car} \Rightarrow_{OBL} \neg\text{damage} \odot \text{compensate} \odot \text{foreclosure} \]

where (i) “$\Rightarrow_{OBL}$” denotes that the conclusion of rule $r_0$ will be treated as an obligation, (ii) The symbol “,” substitutes for and in the antecedents of $r_0$, and (iii) The symbol “$\odot$” replaces the symbol “,” to separate the alternatives (in this case, the reparative obligation that will come in force in case of a violation of a previous element in the chain). Thus, the meaning of rule $r_0$ is that, if an agent drives a car, then she has the obligation not to cause any damage to others; if this happens, she is obliged to compensate; if she fails to compensate, there is an obligation of foreclosure.

Following this perspective, we shall now represent the previous setting with a rule introducing the outcome mode ($OUT$ in the notation of this chapter), that is an outcome rule:

\[ r_1 : \text{Ring\_bearer, destroy\_Ring} \Rightarrow_{OUT} \text{climb\_MistyMts} \odot \text{Gap\_Rohan} \odot \text{pass\_Moria}. \]

In both examples, the sequences express a preference ordering among alternatives. Accordingly, foreclosure and pass\_Moria are not the best options but they are the last (and least) acceptable situations.

We pointed how deeply the factuality of the environment affects what the agent can, or cannot do – no matter how motivated the Fellowship was to climb the Misty Mountains, the sever weather made them eventually give up. We shall model such information by using belief rules ($BEL$ in the notation of this chapter), like

\[ r_2 : \text{severe\_weather} \Rightarrow_{BEL} \neg\text{climb\_MistyMts} \]

meaning that if there are sever weather conditions, then it is not possible (for anyone) to climb them.

In the rest of the section, we shall illustrate the principles and intuitions relating sequences of alternatives (outcome rules), beliefs, obligations, and how
to use them to characterise different types of goal-like attitudes and degrees of commitment to outcomes: desires, goals, intentions, and social intentions.

3.2.1. Desires as acceptable outcomes

Suppose an agent is equipped with the following outcome rules expressing two preference orderings:

\[ r : a_1, \ldots, a_n \Rightarrow_{OUT} b_1 \odot \cdots \odot b_m \]
\[ s : a'_1, \ldots, a'_n \Rightarrow_{OUT} b'_1 \odot \cdots \odot b'_k \]

and that the situations described by \( a_1, \ldots, a_n \) and \( a'_1, \ldots, a'_n \) are mutually compatible but \( b_1 \) and \( b'_1 \) are not, namely \( b_1 = \neg b'_1 \). In this case \( b_1, \ldots, b_m, b'_1, \ldots, b'_k \) are anyway all acceptable outcomes, including the incompatible outcomes \( b_1 \) and \( b'_1 \).

Desires are acceptable outcomes, independently of whether they are compatible with other expected or acceptable outcomes. Let us now contextualise the previous example to better explain the notion of desire by considering the following setting.

Example 1.

\[ F = \{\text{destroy\_Ring, Ring\_bearer, Winter}\} \]
\[ R = \{r_1, r_3 : \text{Winter} \Rightarrow_{OUT} \neg\text{climb\_MistyMts} \odot \text{elf\_scout}\}. \]

The meaning of \( r_3 \) is that Frodo would not consider climbing the Misty Mountains if winter is coming; if he does, then he prefers Legolas to take the lead.

Being that the premises of \( r_1 \) as well as of \( r_3 \) hold, then both rules are activated, and Frodo has both \( \text{climb\_MistyMts} \) and its opposite as acceptable outcomes. Eventually, he must make up his mind.

Notice that if a rule prevails over the other, the elements of the weaker rule which are incompatible with the corresponding ones of the stronger rule are not considered as desires. Suppose that Frodo really prefers to climb the Misty Mountains instead of pass through the Gap of Rohan or Moria. In this setting, \( r_1 \) prevails over \( r_3 \) (\( r_1 > r_3 \) in notation). Given that he explicitly prefers \( r_1 \) to \( r_3 \), his desire is \( \text{climb\_MistyMts} \) and it would be irrational to conclude that he also has the desire \( \neg\text{climb\_MistyMts} \).

Desires have always been the Cinderella of the trinity. Indeed most formal treatments only contain either desires or goals, and often provide a logical representation of goals alone. Even Wooldridge [2000], at the beginning of
3.2. THE INTUITION UNDERNEATH THE FRAMEWORK

the book, describes desires as a self-standing modality and says that they can be mutually exclusive, while in his later formalisation he assumes that they cannot be inconsistent, treating them *de facto* as goals. On the contrary, in our perspective, desires influence the agent’s process of deciding which her goals and intentions are.

Regarding the connections between desires and beliefs we align to the vision of [Dignum et al., 2002]. “It seems that in this respect also there are only weak links. One can certainly desire a situation which one believes impossible, or not desire a situation that eventually will be inevitable, so there are no straightforward relationships between beliefs and desires other than introspection such as $D(\varphi) \Rightarrow B(D(\varphi))$”.

### 3.2.2. Goals as preferred outcomes

We consider a goal as the preferred desire in a chain.

For rule $r$ alone the preferred outcome is $b_1$, and for rule $s$ alone it is $b'_1$. But if both rules are applicable, then a state where both $b_1$ and $b'_1$ hold is not possible: The agent would not be rational if she considers both $b_1$ and $\neg b_1$ as her preferred outcomes. Hence, the agent has to decide if she prefers a state where $b_1$ holds to one where $b'_1$ (i.e., $\neg b_1$) holds, or the other way around. If the agent cannot make up her mind, i.e., she has no way to decide which is the most suitable option for her, then neither the chain of $r$ nor that of $s$ can produce preferred outcomes.

Suppose that the agent opts for the latter option; this can be done if the agent establishes that the second rule overrides the first one ($s > r$). Accordingly, the preferred outcome is $b'_1$ for the chain of outcomes defined by $s$, and $b_2$ is the preferred outcome of $r$. $b_2$ is the second best alternative according to rule $r$: in fact $b_1$ has been discarded as an acceptable outcome given that $s$ prevails over $r$.

Consider now the scenario described by Example 1. In this case, the goal according to $r_1$ is *climb_MistyMts*, while *elf_scout* is the goal for $r_3$.

### 3.2.3. Two degrees of commitment: Intentions and social intentions

The next issue is to clarify which are the acceptable outcomes for an agent to commit to. Naturally, if the agent values some outcomes more than others, she should strive for the best, i.e., for the most preferred outcomes (goals).

We first consider the case where only rule $r$ applies. Here, the agent should commit to the outcome she values the most, i.e., $b_1$. But what if the agent *believes* that $b_1$ cannot be achieved in the environment where she is currently
situated in, or she knows that $\neg b_1$ holds? Committing to $b_1$ would result in a waste of agent’s resources; rationally, she should target the next best outcome, in this case $b_2$. Accordingly, the agent derives $b_2$ as her intention.

An intention is an acceptable outcome which does not conflict with the beliefs describing the environment (the agent has a rational behaviour).

Suppose now that $b_2$ is forbidden, and that the agent is social (an agent is social if the agent would not knowingly commit to anything that is forbidden [Governatori and Rotolo, 2008b]). Once again, in this situation the agent has to lower her expectation and settle for $b_3$, which is considered her social intention.

A social intention is an intention which does not violate any norm governing such an environment (the agent has a social behaviour).

To complete the analysis, consider the situation where both rules $r$ and $s$ apply and, again, the agent prefers $s$ to $r$. As we have seen before, $\neg b_1$ ($b'_1$) and $b_2$ are the preferred outcomes based on the preference of the agent over the two rules. This time we assume that the agent knows she cannot achieve $\neg b_1$ (or equivalently, $b_1$ holds). If the agent is rational, she cannot commit to $\neg b_1$. Thus, the best option for her is to commit to $b'_2$ and $b_1$ (both regarded as intentions and social intentions), where she is guaranteed to be successful.

This scenario reveals a key concept: there are situations where the best course of action for the agent is to commit herself to some outcomes that are not her preferred ones, or even that she would consider not acceptable based only on her preferences, but such that they influence her decision process given that they represent relevant external factors (either her beliefs or the norms that apply to her).

The following example is divided in two parts: The first shows interaction between intentions and beliefs, the latter between social intentions and beliefs/obligations.

Example 2.

\[
F = \{\text{destroy\_Ring, Ring\_bearer, severe\_weather, Winter}\}
\]

\[
R = \{r_1, r_2, r_3\}
\]

\[
> = \{(r_1, r_3)\}.
\]

Saruman has finally used all his magic to summon a blizzard. Even if Frodo has the desire as well as the goal to climb the mountains, the facts in this particular situation force him to form the intention of passing through the Gap of Rohan.
3.2. THE INTUITION UNDERNEATH THE FRAMEWORK

Consider now the following theory

\[ F = \{\text{destroy\_Ring, Gandalf\_wise, Ring\_bearer, severe\_weather, Winter}\} \]
\[ R = \{r_1, r_2, r_3, r_4 : \text{Gandalf\_wise} \Rightarrow \text{OBL} \neg \text{Gap\_Rohan}\} \]
\[ > = \{(r_1, r_3)\}. \]

Fortunately for Frodo, Gandalf is even wiser than powerful. The wizard knows that if the Fellowship would try to pass the Gap of Rohan, it would be almost certain that Saruman would hunt them down and take the Ring for himself. He therefore forbids Frodo to go South. This time, even if Frodo knows that passing the Gap is feasible, the obligation filter prohibits him to do so. Alas, Frodo opts to pass through Dwarrowdelf.

These examples show how different contexts (sets of facts) trigger different rules; in turn, this reflects the fact that the agent may adopt different solutions in order to bring about certain states of affairs depending on a particular context.

Summary and discussion

We started this chapter by setting our research in the context of the BDI literature of cognitive agents. We outlined what we perceived as limits of the current approaches and we moved forward by proposing our solution to the problem of how to formalise in a sole framework the description of the world, the agent’s motivational attitudes, and the norms governing such an environment.

Strength of our proposal being that: (i) The agent is able to express her outcomes through a preference ordering, (ii) Desires, goals, intentions, and social intentions are modalities inter-related with one another as well as with the context (beliefs) and the regulative system (obligations).

We are well aware of that the comparison of our approach against the current literature discussed in Section 3.1 is far from being exhaustive, nor it is needed to be. We shall fill this lack in the related works Section 4.5, after having presented our logical formalisation.
CHAPTER FOUR

OF THE FORMALISATION OF THE LOGIC

Are you a military?
Hell, no. I just like my guns.

\textit{Machine Gun Preacher - Sam Childers}

We shall introduce a modal defeasible logic to model agent’s outcomes, beliefs describing the environment she is acting in, the norms governing such an environment. This will let us characterise situations of compliance with respect both the normative system, and the agent’s outcomes.

Defeasible Logic [Antoniou et al., 2001] is a simple, flexible, and efficient rule based non-monotonic formalism. Its strength lies in its constructive proof theory, which has an argumentation-like structure, and it allows us to draw meaningful conclusions from (potentially) conflicting and incomplete knowledge bases. Being non-monotonic means that more accurate conclusions can be obtained when more pieces of information are given (where some previously derived conclusions no longer follow from the knowledge bases)\footnote{Non-monotonicity allows us to obtain more plausible conclusions when otherwise no conclusion would be obtained at all and this avoids the problem of factual omniscience; for more details see [Governatori et al., 2009b].}.

The framework provided by the proof theory accounts for the possibility of extensions of the logic, in particular extensions with modal operators. Several such extensions have been proposed, which then resulted in successful applications in the area of normative reasoning [Governatori, 2005], modelling agents [Governatori and Rotolo, 2008b, Kravari et al., 2011, Governatori et al., 2009b], and business process compliance [Governatori and Sadiq, 2008, Sadiq and Governatori, 2015]. A model theoretic possible world semantics for modal Defeasible
Logic has been proposed in [Governatori et al., 2012b]. In addition, efficient implementations of the logic (including the modal variants), able to handle very large knowledge bases have been presented in [Lam and Governatori, 2009, Bassiliades et al., 2006, Tachmazidis et al., 2012].

This chapter is based on [Governatori et al., 2011, 2013b].

Outline

The present chapter is structured as follows. Section 4.1 introduces the language adopted and ends by presenting two mechanisms, namely conversion and conflict resolution, to use rules of a given mode to get conclusions of a different one, and to solve conflicts among distinct modalities, respectively.

Section 4.2 outlines the inferential mechanism behind the logic, its main purpose being to establish how to compute factual knowledge, obligations, desires, goals and (social) intentions from existing facts, agent’s primitive motivational attitudes and unconditional obligations.

Finally, Section 4.3 discusses norm and outcome compliances and formally defines them within the logical apparatus we shall propose. In Section 4.4 we introduce the two running examples.

We end this chapter with Section 4.5, where we draw some conclusions but, more importantly, where we confront our framework against the current literature of the field.

4.1. Language

Let PROP be a set of propositional atoms, and MOD = {B, O, D, G, I, SI} the set of modal operators, whose reading is B for belief, O for obligation, D for desire, G for goal, I for intention and SI for social intention. Let Lab be a set of arbitrary labels. The set Lit = PROP ∪ {¬p | p ∈ PROP} denotes the set of literals. The complement of a literal q is denoted by ¬q; if q is a positive literal p, then ¬q is ¬p, and if q is a negative literal ¬p then ¬q is p. The set of modal literals is ModLit = {Xl, ¬Xl | l ∈ Lit, X ∈ {O, D, G, I, SI}}. We assume that modal operator “X” for belief B is the empty modal operator. Accordingly, a modal literal Bl is equivalent to literal l; the complement of B~l and ¬Bl is ¬l.

Definition 1. A defeasible theory D is a structure (F, R, >), where

1. F ⊆ Lit ∪ ModLit is a set of facts or indisputable statements;
2. R contains three sets of rules: for beliefs, obligations, and outcomes;
3. > ∈ \( R \times R \) is a superiority relation to determine the relative strength of conflicting rules.

A theory is finite if the set of facts and rules are so.

Belief rules are used to relate the factual knowledge of an agent, which is her vision of the environment she is situated in, and define the relationships between states of the world. As such, provability for beliefs does not generate modal literals.

Obligation rules determine when and which obligations are in force. The conclusions generated by obligation rules take the O modality.

Finally, outcome rules establish the possible outcomes of an agent depending on the particular context. Apart from obligation rules, outcome rules are used to derive conclusions for all modalities representing goal-like attitudes: «< desires, goals, intentions, and social intentions. Having stated this, outcome rules may generate conclusions with modality D for desire, G for goal, I for intention, and SI for social intention.

Following ideas given of Governatori and Rotolo [2006], rules can gain more expressiveness when a preference operator \( \odot \) is used, whose intuitive meaning is the following. An expression like \( a \odot b \) means that if \( a \) is possible, then \( a \) is the first choice, and \( b \) is the second one; if \( \neg a \) holds, then the first choice is not attainable and \( b \) is the actual choice. This operator is used to build chains of preferences, called \( \odot \)-expressions. The formation rules for \( \odot \)-expressions are:

1. every literal is an \( \odot \)-expression;
2. if \( A \) is an \( \odot \)-expression and \( b \) is a literal then \( A \odot b \) is an \( \odot \)-expression.

In addition, we stipulate the following axioms which describe the behaviour of the operator \( \odot \):

1. \( a \odot (b \odot c) = (a \odot b) \odot c \) (associativity);
2. \( \bigodot_{i=1}^{n} a_i = (\bigodot_{j=1}^{k-1} a_i) \odot (\bigodot_{i=k+1}^{n} a_i) \) where exists \( j \) such that \( a_j = a_k \) and \( j < k \) (duplication and contraction on the right).

\( \odot \)-expressions are given by the agent designer, or obtained through construction rules based on the particular logic [Governatori and Rotolo, 2006].

In this dissertation, we shall exploit the classical definition of defeasible rule in Defeasible Logic [Antoniou et al., 2001], while strict rules and defeaters are omitted\(^2\). A defeasible rule is an expression \( r : A(r) \Rightarrow_{X} C(r) \), where

\(^2\)The restriction does not result in any loss of generality: (i) The superiority relation does not play any role in proving definite conclusions, and (ii) For defeasible conclusions, Antoniou et al. [2001] prove that it is always possible to remove a) strict rules from the superiority
1. \( r \in \text{Lab} \) is the name of the rule;
2. \( A(r) = \{a_1, \ldots, a_n\} \), the antecedent of the rule, is the set of the premises of the rule. Each \( a_i \) is either a literal or a modal literal;
3. \( X \in \{B,O,U\} \) represents the mode of the rule: \( \Rightarrow_B \), \( \Rightarrow_O \), \( \Rightarrow_U \) denote respectively rules for beliefs, obligations, and outcomes (from now on, we omit the subscript B in rules for beliefs, i.e., \( \Rightarrow \) is used as a shortcut for \( \Rightarrow_B \));
4. \( C(r) \) is the consequent (or head) of the rule, which is a single literal if \( X = B \), and an \( \circ \)-expression otherwise\(^3\).

We shall use the following abbreviations on (sets of) rules: \( R^X \) (\( R^X[q] \)) denotes all rules of mode \( X \) (with consequent \( q \)), and \( R[q] = \bigcup_{X \in \{B,O,U\}} R^X[q] \). With \( R[q,i] \) we denote the set of rules whose head is \( \circ_{j=1}^n c_j \) and \( c_i = q \), with \( 1 \leq i \leq n \). Finally, \( (l,i) \in C(r) \) is a shortcut denoting that literal \( l \) appears at index \( i \) in the consequent of rule \( r \).

Notice that labelling the rules of Defeasible Logic produces nothing more than a simple treatment of the modalities, so that two interaction strategies between modal operators are analysed by Governatori and Rotolo [2008b]: rule conversion and conflict resolution.

In the remainder, we shall define a completely new inference machinery that takes this into account by adding preferences and dealing with a larger set of modalised conclusions which are not necessarily obtained from the corresponding rules, but also by using other rule types. For example, we argued in Section 3.2 that a goal can be viewed as a preferred outcome and so the fact that a certain goal \( Gp \) is derived depends upon whether we can obtain \( p \) a preferred using a rule for \( U \).

### 4.1.1. Rule conversion

It is sometimes meaningful to use rules for a modality \( X \) as they were for another modality \( Y \), i.e., to convert one type of conclusions into a different one.

Let us consider the following two statements: “On weekends, I have the intention to go surfing” and “When you surf, you go to the ocean”. We can formalise the first statements with a defeasible outcome rule and the second one

\(^3\)It is worth noting that modal literals can occur only in the antecedent of rules: the reason is that rules are used to derive modal conclusions and we do not conceptually need to iterate modalities. The motivation of a single literal as a consequent for belief rules is dictated by the intended reading of the belief rules, where such rules are used to describe the environment.
with a defeasible belief rule as

\[
\text{weekend} \Rightarrow \cup \text{surfing} \\
\text{surfing} \Rightarrow B \text{go}_\text{ocean}
\]

If today is a weekend day, we may conclude that the desire of surfing implies the desire of going to the ocean. This is done in two steps: (i) From the outcome rule we derive the desire of surfing (D_{surfing}); (ii) By using the belief rule with the premise D_{surfing}, we derive D_{go_\text{ocean}}. Notice that, in this particular example, from the outcome rule we would also obtain the goal and the intention of surfing and, consequently, the goal and the intention of going to the ocean.

Formally, we define an asymmetric binary relation \(\text{Convert} \subseteq \text{MOD} \times \text{MOD}\) such that \(\text{Convert}(X, Y)\) means ‘a rule of mode \(X\) can be used also to produce conclusions of mode \(Y\)’. This intuitively corresponds to the following logical schema:

\[
Y a_1, \ldots, Y a_n \quad a_1, \ldots, a_n \Rightarrow X b \\
Y b \Rightarrow \text{Convert}(X, Y).
\]

In our framework obligations and goal-like attitudes cannot change what the agent believes or how she perceives the world, thus we only consider conversion from beliefs to the other modes (i.e., \(\text{Convert}(B, X)\) with \(X \in \{O, D, G, I, SI\}\)). Accordingly, we enrich the notation with \(R^B_X\), denoting the set of belief rules that can be used for a conversion to mode \(X \in \text{MOD} \setminus \{B\}\). The antecedent of all such rules must be not empty, and not contain any modal literal.

**Example 3.** We reframe the theory of Example 1.

\[
F = \{D_{\text{give}_\text{Ring}_\text{Gandalf}}\} \\
R = \{r_5 : D_{\text{give}_\text{Ring}_\text{Gandalf}} \Rightarrow \neg D_{\text{Ring}_\text{bearer}}\} \\
> = \emptyset.
\]

where we stipulate that \(\text{Convert}(B, D)\) holds.

Frodo desires to entrust Gandalf with The Ring. If he does so, bearing The Ring will no longer be his duty. Thus, we are allowed to state that his desire to give The Ring to Gandalf implies the desire to no longer be the Ring bearer. This is the case since we can use rule \(r_5\) to convert beliefs into desires.

It is important to notice that the derivation of \(D_{\text{give}_\text{Ring}_\text{Gandalf}}\) as belief is not necessary to use rule \(r_5\) via conversion to derive (in this case) a desire. What it does matter is to derive the premises of the rule as desires.
4.1.2. Conflict-detection/resolution

It is crucial to identify criteria for detecting and solving conflicts between different modalities. We define an asymmetric binary relation $\text{Conflict} \subseteq \text{MOD} \times \text{MOD}$ such that $\text{Conflict}(X, Y)$ means ‘modes $X$ and $Y$ are in conflict and mode $X$ prevails over $Y$’. In our framework, we consider conflicts between: beliefs over intentions, beliefs over social intentions, and obligations over social intentions. Accordingly, the agents we consider are characterised by:

- $\text{Conflict}(B, I)$, $\text{Conflict}(B, SI)$ meaning that agents are realistic (cf. [Broersen et al., 2002]), and
- $\text{Conflict}(O, SI)$ meaning that agents are social (cf. [Governatori and Rotolo, 2008b]).

Consider the scenario in Example 2 with $\text{Conflict}(B, I)$ and $\text{Conflict}(O, SI)$. We recall that rule $r_4$ states the prohibition to pass South via the Gap of Rohan. Thus, Frodo has the intention to pass South through the Gap of Rohan, but he does not have the social intention to do so. This follows since rule $r_4$ prevents through conflict to prove $SI\_\text{Gap\_Rohan}$. At the end, it is up to the agent (or the designer of the agent) whether to comply with the obligation, or not.

There are two applications of the superiority relation: The first considers rules of the same mode while the latter compares rules of different mode. Given $r \in R^X$ and $s \in R^Y$, $r > s$ iff $r$ converts $X$ into $Y$, or $s$ converts $Y$ into $X$, i.e., the superiority relation is used when rules, each with a different mode, are used to produce complementary conclusions of the same mode.

**Example 4.** Consider the following theory:

$$F = \{\text{little\_hobbit}, \text{other\_ways\_impracticable}\},$$

$$R = \{r_6 : \text{pass\_Moria} \Rightarrow \text{face\_orcs},$$

$$r_7 : \text{other\_ways\_impracticable} \Rightarrow \text{pass\_Moria},$$

$$r_8 : \text{little\_hobbit} \Rightarrow \text{U} \text{\neg face\_orcs}\},$$

$$> = \{(r_6, r_8)\},$$

where we stipulate that $\text{Convert}(B, I)$ holds.

It so happens that Frodo is a little hobbit. As such, he does not like the idea of fighting hordes of orcs. Alas, all the other ways are impracticable so he intends to lead the Fellowship through the gates of Moria. Passing through Dwarrowdelf means that they will almost certainly face orcs. We may conclude that Frodo intends to face the
4.2. Inferential Mechanism

A proof $P$ of length $n$ is a finite sequence $P(1),\ldots,P(n)$ of tagged literals of the type $+\partial_X q$ and $-\partial_X q$, where $X \in$ MOD. The proof conditions below define the logical meaning of such tagged literals. As a conventional notation, $P(1..i)$ denotes the initial part of the sequence $P$ of length $i$. Given a modal defeasible theory $D$ and a proof $P$ in it, a literal $q$ is defeasibly provable in $D$ at line $n$ with the mode $X$ if there is a line $P(m), n \leq m$, such that $P(m) = +\partial_X q$. A literal $q$ is not provable, or refuted, in $D$ at line $n$ with the mode $X$ if there is a line $P(m), n \leq m$, such that $P(m) = -\partial_X q$. From now on, we shall use $D + \partial_X q$ iff there is a proof $P$ in $D$ such that $P(m) = +\partial_X q$ for an index $m$.

Given $\# \in \{\Delta, \partial\}$ and a proof $P$ in $D$, a literal $q$ is $\#$-provable at line $n$ in $D$ if there is a line $P(m), n \leq m$, of $P$ such that $P(m) = +\# q$. A literal $q$ is $\#$-rejected at line $n$ in $D$ if there is a line $P(m), n \leq m$, of $P$ such that $P(m) = -\# q$. When clear from the context, we simply use provable and rejected.

In order to characterise the notions of provability/refutability for beliefs ($\pm \partial_B$), obligations ($\pm \partial_O$), desires ($\pm \partial_D$), goals ($\pm \partial_G$), intentions ($\pm \partial_I$) and social intentions ($\pm \partial_SI$), it is essential to define when a rule is applicable or discarded. To this end, the preliminary notions of body-applicable and body-discarded must be introduced, stating that a rule is body-applicable when each literal in its body is proved with the suitable modality, or that a rule is body-discarded if (at least) one of its premises has been refuted.

**Definition 3.** Let $P$ be a proof and $X \in \{O,D,G,I,SI\}$. A rule $r \in R$ is body-applicable (at $P(n+1)$) iff for all $a_i \in A(r)$:

1. if $a_i = X l$ then $+\partial_X l \in P(1..n)$,
2. If \( a_i = \neg Xl \) then \( \neg \partial_X l \in P(1..n) \),
3. If \( a_i = \partial_X l \) then \( + \partial_X l \in P(1..n) \).

**Definition 4.** Let \( P \) be a proof and \( X \in \{O, D, G, I, SI\} \). A rule \( r \in R \) is body-discard
ed (at \( P(n+1) \)) iff there is \( a_i \in A(r) \) such that
1. \( a_i = Xl \) and \( \neg \partial_X l \in P(1..n) \), or
2. \( a_i = \neg Xl \) and \( + \partial_X l \in P(1..n) \), or
3. \( a_i = l \in \text{Lit} \) and \( \neg \partial_B l \in P(1..n) \).

As already stated, belief rules allow us to derive literals with di-
erent modalities through the conversion apparatus. The applicability mechanism must take
into account this constraint.

**Definition 5.** Let \( P \) be a proof. A rule \( r \in R \) is Conv-applicable (at \( P(n+1) \)) for \( X \) iff
1. \( r \in R^B \),
2. \( A(r) \neq \emptyset \),
3. \( A(r) \cap \text{ModLit} = \emptyset \), and
4. \( \forall a \in A(r), + \partial_X a \in P(1..n) \).

**Definition 6.** Let \( P \) be a proof. A rule \( r \in R \) is Conv-discarded (at \( P(n+1) \)) for \( X \) iff
1. \( r \notin R^B \), or
2. \( A(r) = \emptyset \), or
3. \( A(r) \cap \text{ModLit} \neq \emptyset \), or
4. \( \exists a \in A(r) \text{ s.t. } \neg \partial_X a \in P(1..n) \).

**Example 5.** Let us consider the following theory

\[
F = \{a, b, Oc\},
\]
\[
R = \{r_1 : a \Rightarrow O b, r_2 : b, c \Rightarrow d\},
\]
\[
> = \emptyset.
\]

\( r_1 \) is applicable while \( r_2 \) is not, since \( c \) is not proved as a belief. Instead, \( r_2 \) is
Conv-applicable for \( O \), since \( Oc \) is a fact and \( r_1 \) gives \( Ob \).

The notion of applicability gives guidelines on how to consider the next
element in a given chain. Hereafter we report an analysis for each specific type
of rule.

Beliefs: Since a rule for belief cannot generate reparative chains but only single
literals, we conclude that the applicability condition for belief collapses
into body-applicability.
Obligations: Each element before the current one must be a violated obligation.

Desires: Given that each element in an outcome chain represents a possible desire, we only require the rule to be applicable either directly, or through Convert relation.

Goals: A literal is a candidate to be a goal only if none of the previous elements in the chain has been proved as such.

Intentions: An intention must pass the wishful thinking filter, that is there is no factual knowledge for the opposite conclusion.

Social Intentions: It is an intention which is also constrained not to violate any norm, that is the literal passes the sociality filtering and the opposite literal has not been derived as an obligation.

Follows the formalisations of being applicable and discarded.

**Definition 7.** Given a proof $P$, $r \in R[q, i]$ is applicable (at index $i$ and $P(n + 1)$) for

1. B iff $r \in R^B$ and is body-applicable.
2. O iff either
   - (2.1) (2.1.1) $r \in R^O$ and is body-applicable,
     - (2.1.2) $\forall c_k \in C(r), \ k < i, +\partial O c_k \in P(1..n) \ and \ -\partial c_k \in P(1..n)$, or
   - (2.2) $r$ is Conv-applicable.
3. D iff either
   - (3.1) $r \in R^U$ and is body-applicable, or
   - (3.2) Conv-applicable.
4. $X \in \{G, I, SI\}$ iff either
   - (4.1) (4.1.1) $r \in R^U$ and is body-applicable, and
     - (4.1.2) $\forall c_k \in C(r), \ k < i, +\partial Y \sim c_k \in P(1..n)$ for some $Y$ s.t. Conflict$(Y, X)$ and $-\partial X c_k \in P(1..n)$, or
   - (4.2) $r$ is Conv-applicable.

For $G$ there are no conflicts; for $I$ we have Conflict$(B, I)$, and for $SI$ we have Conflict$(B, SI)$ and Conflict$(O, SI)$. Condition (4.1.2) may appear redundant while stating that we need both $+\partial Y c_k$ as well as $-\partial X c_k$. Would not it be sufficient to have only $+\partial Y c_k$? The answer is, naturally, it is not. This is so because another rule may prove $+\partial X c_k$ and, in that case, we do not want to proceed in the chain.
Definition 8. Given a proof $P$, $r \in R[q, i]$ is discarded (at index $i$ and $P(n + 1)$) for

1. B if $r \in R^B$ and is body-discarded.
2. O if
   (2.1) (2.1.1) $r \notin R^O$ or is body-discarded, or
   (2.1.2) $\exists c_k \in C(r)$, $k < i$, s.t. $-\partial c_k \in P(1..n)$ or $\partial c_k \in P(1..n)$, and
   (2.2) $r$ is Conv-discarded.
3. D if
   (3.1) $r \notin R^U$ or is body-discarded, and
   (3.2) Conv-discarded.
4. $X \in \{G, I, SI\}$ if
   (4.1) (4.1.1) $r \notin R^U$ or is body-discarded, or
   (4.1.2) $\exists c_k \in C(r)$, $k < i$, s.t. $-\partial c_k \in P(1..n)$ for all $Y$
   s.t. Conflict($Y$, $X$) or $\partial c_k \in P(1..n)$ and
   (4.2) $r$ is Conv-discarded.

For $G$ there are no conflicts; for $I$ we have Conflict($B, I$), and for $SI$ we have Conflict($B, SI$) and Conflict($O, SI$).

Notice that conditions of Definition 8 are the strong negation\textsuperscript{4} of those in Definition 7. The conditions to establish a rule being discarded correspond to the constructive failure to prove that the same rule is applicable.

We are now ready to define the proof conditions for the modal operators given in this chapter. We start with that for desire.

Definition 9. The proof conditions of defeasible provability for desire are

$+\partial_D$: If $P(n + 1) = +\partial_D q$ then

(1) $Dq \in F$ or
(2) (2.1) $-Dq \notin F$ and
   (2.2) $\exists r \in R[q, i]$ s.t. $r$ is applicable for $D$ and
   (2.3) $\forall s \in R[~q, j]$ either
      (2.3.1) $s$ is discarded for $D$, or
      (2.3.2) $s \neq r$.

The above conditions determine when we are able to assert that $q$ is a desire. Specifically, a desire is either fact of the theory (1), or there is an outcome rule applicable for $D$ (2.2) for which there is no stronger argument for the opposite desire (2.3). Notice that condition (2.1) does not impose that $D \sim q \notin F$ but\textsuperscript{4}

\textsuperscript{4}The strong negation principle is closely related to the function that simplifies a formula by moving all negations to an inner most position in the resulting formula, and replaces the positive tags with the respective negative tags, and the other way around, see [Antoniou et al., 2000, Governatori et al., 2009b].
instead we have $\neg Dq \notin F$. This is such because expressing opposite desires makes the theory still coherent which is not in the case of $\neg Dq \in F$.

The negative counterpart $\neg Dq$ is obtained by the principle of strong negation.

**Definition 10.** The proof conditions of defeasible refutability for desire are $\neg Dq$: If $P(n + 1) = \neg Dq$ then

1. $Dq \notin F$ and
2. (2.1) $\neg Dq \in F$, or
   2.2. $\forall r \in R[q, i]$ either $r$ is discarded for D, or
   2.3. $\exists s \in R[\neg q, j]$ s.t.
      2.3.1. $s$ is applicable for D and
      2.3.2. $s > r$.

Literal $q$ is refuted as desire whenever is not a fact (1) and either the agent directly expresses that she does not want it to be so (2.1), or there exists an outcome rule applicable for D which is stronger (2.3).

The proof conditions for $+\partial_X$, with $X \in \text{MOD} \setminus \{D\}$ are as follows:

**Definition 11.** The proof conditions of defeasible provability for $X \in \text{MOD} \setminus \{D\}$ are $+\partial_X$: If $P(n + 1) = +\partial_X q$ then

1. $Xq \in F$ or
2. (2.1) $\neg Xq \notin F$ and $(Y \neg q \notin F \text{ for } Y = X \text{ or Conflict}(Y, X))$ and
   2.2. $\exists r \in R[q, i]$ s.t. $r$ is applicable for $X$ and
   2.3. $\forall s \in R[\neg q, j]$ either
      2.3.1. $Y s.t. Y = X \text{ or Conflict}(Y, X)$, $s$ is discarded for $X$, or
      2.3.2. $\exists t \in R[q, k]$ s.t. $t$ is applicable for $T$ and either
         2.3.2.1. $t > s \text{ if } Y = T, \text{ Convert}(Y, T)$, or $\text{Convert}(T, Y)$; or
         2.3.2.2. $\text{Conflict}(T, Y)$.

To show that a literal $q$ is defeasibly provable with the modality $X$ we have two choices: (1) modal literal $Xq$ is a fact; or (2) we need to argue using the defeasible part of $D$. For (2), we require that (2.1) a complementary literal (of the same modality, or of a conflictual modality) does not appear in the set of facts, and (2.2) there must be an applicable rule for $\pm \partial_X$ and $q$. Moreover, each possible attack brought by a rule $s$ for $\neg q$ has to be either discarded (2.3.1), or successfully counterattacked by another stronger rule $t$ for $q$ (2.3.2). We recall that the superiority relation combines rules of the same mode, rules with different modes that produce complementary conclusion of the same mode through conversion (both considered in clause (2.3.2.1)), and conflictual modalities (clause 2.3.2.2). Obviously, if $X = B$ then the proof conditions reduce to those of classical defeasible logic [Antoniou et al., 2001].
Again, conditions for $-\partial_X$ are derived by the principle of strong negation from that for $+\partial_X$ and are as follows:

**Definition 12.** The proof conditions of defeasible refutability for $X \in \{O, G, I, SI\}$ are

$-\partial_X$: If $P(n + 1) = -\partial_X q$ then

1. $Xq \notin F$ and

2. $(2.1) \neg Xq \in F$ or ($Y \leftarrow q \in F$, for $Y = X$ or Conflict($Y, X$)) or

   (2.2) $\forall r \in R[q, i]$ either $r$ is discarded for $X$ or

   (2.3) $\exists s \in R[\neg q, j]$ s.t.

   (2.3.1) $\exists Y$ s.t. ($Y = X$ or Conflict($Y, X$)) and $s$ is applicable for $Y$, and

   (2.3.2) $\forall T, \forall t \in R[q, k]$ either $t$ is discarded for $T$ or

   (2.3.2.1) $t \nless s$ if $Y = T$, Convert($Y, T$), or Convert($T, Y$); and

   (2.3.2.2) not Conflict($T, Y$).

To refute a literal $q$ with modality $X$, it must not be a fact of the theory (1). Also, either an opposite (conflicting) literal is a fact of the theory (2.1), or every rule for $X$ is discarded (2.2), or (2.3) there is a rule for a possibly conflicting mode which is applicable and not defeated (2.3).

To better understand how applicability and proof conditions interact to define the (defeasible) conclusions of a given theory, we consider the example below.

**Example 6.** Let $D$ be the following modal theory

$$F = \{a_1, a_2, \neg b_1, O\neg b_2\},$$

$$R = \{r : a_1 \Rightarrow b_1 \odot b_2 \odot b_3 \odot b_4,$$

$$s : a_2 \Rightarrow b_4\},$$

$$> = \emptyset.$$  

Here, $r$ is trivially applicable for $D$ and $+\partial_3 b_1$, holds, for $1 \leq i \leq 4$. Moreover, we have $+\partial_G b_1$ and $r$ is discarded for $G$ after $b_1$. Since $+\partial \neg b_1$, it follows that $-\partial b_1$ holds (as well as $-\partial_S b_1$); the rule is applicable for $I$ and $b_2$, and we are able to prove $+\partial b_2$; thus the rule becomes discarded for $I$ and $b_3$ as well as $b_4$. Given that $O\neg b_2$ is a fact, $r$ is discarded for $SI$ and $b_2$ resulting in $-\partial_S b_2$, which in turn makes the rule applicable for $SI$ and $b_3$, proving $+\partial_S b_3$. As we have argued before, this makes $r$ discarded for $b_4$. Nevertheless, even if $r$ is discarded for $SI$ and $b_4$, we have $D \vdash +\partial_S b_4$ following by $s$; specifically, $D \vdash +\partial_X b_4$ with $X \in \{D, G, I, SI\}$ given that $s$ is trivially applicable for $X$.

For a further illustration of how the derivation machinery works and handles the preference relation as well as Convert and Conflict relations, let us consider Condition (2.3.2) of Definition 11; Table 4.1 reports all the scenarios where
4.2. INFERENTIAL MECHANISM

a rule $r$ proves $+\delta_{SI}q$ when attacked by an applicable rule $s$, which in turn is successfully counterattacked by an applicable rule $t$. For the sake of clarity, notation $B, X$ (with $X \in \{O, SI\}$) denotes that the belief rule is Conv-applicable for $X$. For example, let us examine the sixth row. If the outcome rule $r$ is applicable for mode $SI$, then a counterattack may come from a rule $s$ for obligation which, in turn, may be defeated by a rule either for belief, or for obligation.

<table>
<thead>
<tr>
<th>Mode of $r$</th>
<th>Mode of $s$</th>
<th>Mode of $t$</th>
<th>$+\delta_{SI}q$ because...</th>
</tr>
</thead>
<tbody>
<tr>
<td>U applicable for SI</td>
<td>U applicable for SI</td>
<td>U applicable for SI</td>
<td>$t &gt; s$</td>
</tr>
<tr>
<td>U applicable for SI</td>
<td>U applicable for SI</td>
<td>O</td>
<td>Conflict(O, SI)</td>
</tr>
<tr>
<td>U applicable for SI</td>
<td>U applicable for SI</td>
<td>B, SI</td>
<td>$t &gt; s$</td>
</tr>
<tr>
<td>U applicable for SI</td>
<td>U applicable for SI</td>
<td>B, O</td>
<td>Conflict(O, SI)</td>
</tr>
<tr>
<td>U applicable for SI</td>
<td>O</td>
<td>O</td>
<td>$t &gt; s$</td>
</tr>
<tr>
<td>U applicable for SI</td>
<td>O</td>
<td>B, O</td>
<td>$t &gt; s$</td>
</tr>
<tr>
<td>U applicable for SI</td>
<td>B, SI</td>
<td>U applicable for SI</td>
<td>$t &gt; s$</td>
</tr>
<tr>
<td>U applicable for SI</td>
<td>B, SI</td>
<td>O</td>
<td>Conflict(O, SI)</td>
</tr>
<tr>
<td>U applicable for SI</td>
<td>B, SI</td>
<td>B, SI</td>
<td>$t &gt; s$</td>
</tr>
<tr>
<td>U applicable for SI</td>
<td>B, SI</td>
<td>B, O</td>
<td>Conflict(O, SI)</td>
</tr>
<tr>
<td>U applicable for SI</td>
<td>B, O</td>
<td>B, O</td>
<td>$t &gt; s$</td>
</tr>
<tr>
<td>B, SI</td>
<td>U applicable for SI</td>
<td>U applicable for SI</td>
<td>$t &gt; s$</td>
</tr>
<tr>
<td>B, SI</td>
<td>U applicable for SI</td>
<td>O</td>
<td>Conflict(O, SI)</td>
</tr>
<tr>
<td>B, SI</td>
<td>U applicable for SI</td>
<td>B, SI</td>
<td>$t &gt; s$</td>
</tr>
<tr>
<td>B, SI</td>
<td>U applicable for SI</td>
<td>B, O</td>
<td>Conflict(O, SI)</td>
</tr>
<tr>
<td>B, SI</td>
<td>O</td>
<td>O</td>
<td>$t &gt; s$</td>
</tr>
<tr>
<td>B, SI</td>
<td>O</td>
<td>B, O</td>
<td>$t &gt; s$</td>
</tr>
<tr>
<td>B, SI</td>
<td>B, SI</td>
<td>U applicable for SI</td>
<td>$t &gt; s$</td>
</tr>
<tr>
<td>B, SI</td>
<td>B, SI</td>
<td>O</td>
<td>Conflict(O, SI)</td>
</tr>
<tr>
<td>B, SI</td>
<td>B, SI</td>
<td>B, SI</td>
<td>$t &gt; s$</td>
</tr>
<tr>
<td>B, SI</td>
<td>B, SI</td>
<td>B, O</td>
<td>Conflict(O, SI)</td>
</tr>
<tr>
<td>B, SI</td>
<td>B, O</td>
<td>B, O</td>
<td>$t &gt; s$</td>
</tr>
</tbody>
</table>

**Table 4.1.** Definition 11, Condition (2.3.2): Attacks and counterattacks

The next definition extends the concept of complement for modal literals and is used to establish the logical connection among proved and refuted literals in our framework.

**Definition 13.** The complement of a given modal literal $l$, denoted by $\overline{l}$, is:

1. if $l = Dm$, then $\overline{l} = \{\neg Dm\}$;
2. if $l = Xm$, then $\overline{l} = \{\neg Xm, X\neg m\}$, with $X \in \{O, G, I, SI\}$;
3. if $l = \neg Xm$, then $\overline{l} = \{Xm\}$.

The logic resulting from the above proof conditions enjoys properties describing the appropriate behaviour of the modal operators when the underlying
theory is deemed to be consistent.

**Definition 14.** A modal defeasible theory $D = (F, R, >)$ is consistent iff $> \text{ is acyclic and } F \text{ does not contain pairs of complementary (modal) literals, that is if } F \text{ does not contain pairs like (i) } l \text{ and } \neg l, \text{ (ii) } Xl \text{ and } \neg Xl \text{ with } X \in \text{MOD, and (iii) } Xl \text{ and } X\neg l \text{ with } X \in \{G, I, SI\}.$

**Proposition 15.** Let $D$ be a consistent, finite modal defeasible theory. For any literal $l$, it is not possible to have both

1. $D \vdash +\partial_X l$ and $D \vdash -\partial_X l$ with $X \in \text{MOD};$
2. $D \vdash +\partial_X l$ and $D \vdash +\partial_X \neg l$ with $X \in \text{MOD} \setminus \{D\}.$

**Proof.**

1. *(Coherency of the logic)* The negative proof tags are the strong negation of the positive ones, and so are the conditions of a rule being discarded (Definition 8) for a rule being applicable (Definition 7). Hence, when the conditions for $+\partial_X$ hold, those for $-\partial_X$ do not.

2. *(Consistency of the logic)* We split the proof in two cases: (i) at least one of $Xl$ and $X\neg l$ is in $F$, and (ii) none of them is in $F$. For (i) the proposition immediately follows by the assumption of consistency. In fact, suppose that $Xl \in F$. Then clause (1) of $+\partial_X$ holds for $l$. By consistency $X\neg l \notin F$, thus clause (1) of Definition 11 does not hold for $\neg l$. Since $Xl \in F$, also clause (2.1) is always falsified for $\neg l$, and the thesis is proved.

For (ii), let us assume that both $+\partial_X l$ and $+\partial_X \neg l$ hold in $D$. A straightforward assumption derived by Definitions 7 and 8 is that no rule can be at the same time applicable and discarded for the derivation of $\pm \partial_X$ for any literal $l$ and its complement. Thus, we have that there are applicable rules for both $l$ and $\neg l$ for $\pm \partial_X$. This means that clause (2.3.2) of Definition 11 holds for both $l$ and $\neg l$. Therefore, for every applicable rule for $l$ there is an applicable rule for $\neg l$ stronger than the rule for $l$. Symmetrically, for every applicable rule for $\neg l$ there is an applicable rule for $l$ stronger than the rule for $\neg l$. Since the set of rules in $D$ is finite by construction, this situation is possible only if there is a cycle in the transitive closure of the superiority relation, which is in contradiction with the hypothesis of $D$ being consistent. $\square$

The meaning of the above proposition being that, for instance, it is not possible for an agent to obey something that is, at the same time, obligatory and not obligatory, or that she cannot commit herself in doing something and, at the same time, she does not. Dissimilar is for desires when she may have opposite desires given different situations, but then she will be able to plan for only one between the two alternatives.
4.2. INFERENTIAL MECHANISM

The next proposition governs the interactions between different modalities, as well as the relationship between proved literals and refuted complementary literals of the same modality.

**Proposition 16.** Let $D$ be a consistent modal defeasible theory. For any literal $l$, the following statements hold:

1. if $D \vdash +\partial_X l$, then $D \vdash -\partial_X \neg l$ with $X \in \text{MOD} \setminus \{D\}$;  
2. if $D \vdash +\partial l$, then $D \vdash -\partial \neg l$;  
3. if $D \vdash +\partial l$ or $D \vdash +\partial_D l$, then $D \vdash -\partial_{SI} l$.  
4. if $D \vdash +\partial_G l$, then $D \vdash +\partial_D l$  
5. if $D \vdash +\partial_{SI} l$, then $D \vdash +\partial_D l$.

**Proof.** Case 1. Let $D$ be a consistent modal defeasible theory, and $D \vdash +\partial_X l$. Literal $\neg l$ can be in only one of the following, mutually exclusive situations: (i) $D \vdash +\partial_X \neg l$; (ii) $D \vdash -\partial_X \neg l$; (iii) $D \not\vdash +\partial_X \neg l$. Proposition 15 part 2 allows us to exclude case (i), since $D \vdash +\partial_X l$ by hypothesis. Case (iii) denotes situations where there are loops in the theory involving literal $\neg l$, but inevitably this would affect also the provability of $Xl$, i.e., we would not be able to give a proof for $+\partial_X l$ as well. This is in contradiction with the hypothesis. Consequently, situation (ii) must be the case.

Cases 2.–3. directly follow by Definitions 7 and 8.  
Definitions 7 and 11 justify Case 4., given that $G$ is not involved in any Conflict relation; the same definitions justify Case 5.  

Trivially, from 4. $-\partial_D l$ implies $-\partial_G l$. On the contrary, the following statements do not hold:

6. if $D \vdash +\partial_D l$, then $D \vdash +\partial_X l$ with $X \in \{G, I, SI\}$;  
7. if $D \vdash +\partial_G l$, then $D \vdash +\partial_X l$ with $X \in \{I, SI\}$;  
8. if $D \vdash +\partial_X l$, then $D \vdash +\partial_Y l$ with $X = \{I, SI\}$ and $Y = \{D, G\}$;  
9. if $D \vdash -\partial_Y l$, then $D \vdash -\partial_X l$ with $Y \in \{D, G\}$ and $X \in \{I, SI\}$;  
10. if $D \vdash -\partial_G l$, then $D \vdash -\partial_X l$ with $X \in \{I, SI\}$;  
11. if $D \vdash -\partial_D l$, then $D \vdash -\partial_{SI} l$.

Statements 6. and 7. directly follow by Definitions from 7 to 12 and lay on the intuitions presented in Section 3.2. Statements 8. and 9. reveal the true nature of expressing outcomes in a preference order: it may be the case that the agent desires something (may it be even her preferred outcome) but if the factuality of the environment makes this outcome impossible to reach, then she should

---

5For example, situations like $X\neg l \Rightarrow_X \neg l$, where the proof conditions will generate a loop without introducing a proof.
not pursue such an outcome, and instead committing herself on the next option available. Statements 10. and 11. are trivial by Definitions 7 and 8.

The first four statements of Proposition 16 exhibit a common feature which can be illustrated by the common idiom: “What’s your plan B?”. The meaning is: even if you are willing for an option, if such an option is not feasible you need to strive for the plan B (as many action movies prove).

Example 2 in the extended version (reported below for the sake of clarity) offers counterexamples showing the reason why the above statements do not hold.

\[ F = \{ \text{destroy\_Ring, Ring\_bearer, severe\_weather, Winter} \} \]
\[ R = \{ r_1: \text{Ring\_bearer, destroy\_Ring} \Rightarrow U \text{ climb\_MistyMts} \odot \text{ Gap\_Rohan} \odot \text{ pass\_Moria}, \]
\[ r_2: \text{severe\_weather} \Rightarrow_{BEL} \neg \text{climb\_MistyMts}, \]
\[ r_3: \text{Winter} \Rightarrow U \neg \text{climb\_MistyMts} \odot \text{elf\_scout} \}
\[ > = \{(r_1, r_3)\}. \]

Given \( r_1 > r_3 \), Frodo has the desire to \( \text{climb\_MistyMts} \), and this is also his preferred outcome, while \( r_3 \) produces \( \text{elf\_scout} \) as preferred outcome. Nonetheless, given the derivation of \( \neg \text{climb\_MistyMts} \) as a belief due to \( r_2 \), this is not his intention, while so are \( \text{Gap\_Rohan} \) and \( \neg \text{climb\_MistyMts} \). Notice that a rule may contribute to derive a literal with modality I but not G as is the case for rule \( r_3 \) and literal \( \neg \text{climb\_MistyMts} \).

### 4.3. Norm and outcome compliance

Our logic is able to model in a natural way the concepts of being compliant with respect to norms and outcomes.

Consider, for instance, the obligation rule \( r: \Gamma \Rightarrow_O o_1 \odot o_2 \odot o_3 \) in a theory where \( Oo_1 \) and \( Oo_2 \) are the case, but not \( Oo_3 \). To be compliant with \( r \), the agent has either to prove \( Bo_1 \), or to compensate the former violation of \( Oo_1 \) by deriving \( Bo_2 \). If the agent fails in obtaining \( Bo_2 \) as well, she will have no more means to be compliant. In fact, even the derivation of \( Bo_3 \) cannot help her in being compliant with \( r \), given that \( Oo_3 \) is not in force in the actual normative system.

To formalise the concept of compliance, we first introduce the new literal \( \perp \)
that is interpreted in a not compliant situation, and we provide proof conditions to (defeasibly) derive it. We exploit the modal derivations of \( \bot \) to formally characterise norm compliant \((-\partial_O \bot)\) and outcome compliant \((-\partial_X \bot, X \in \{G, I, SI\})\) situations.

**Definition 17** (Norm compliance). *Given a modal, defeasible theory \( D = (F, R, >) \), \( D \) is norm compliant (with respect mode \( O \)) if, and only if \( D \vdash -\partial_O \bot \), where -\( \partial_O \bot \): If \( P(n + 1) = -\partial_O \bot \) then

1. \( \forall r \in R^O \cup R^{BO} \) either
   1. \( r \) is discarded, or
   2. \( -\partial_O c_1 \in P(1..n) \), or
   3. \( \exists c_i \in C(r), 1 \leq i \leq j, s.t. +\partial_O c_i \in P(1..n) \) and \( +\partial c_i \in P(1..n) \).

To be norm compliant, all applicable rules producing an obligation are such that either all elements in the consequent are not actually active obligations (condition (2.1)), or one element \( c_i \) is an obligation in force and is fulfilled (condition (2.2)).

The situation is slightly different when addressing outcome compliance.

**Definition 18** (Outcome compliance). *Given a modal, defeasible theory \( D = (F, R, >) \), \( D \) is outcome compliant with respect mode \( X \) if, and only if \( D \vdash -\partial_X \bot, \) where -\( \partial_X \bot \): If \( P(n + 1) = -\partial_X \bot \) then

1. \( \forall r \in R^U \cup R^{BX}, \) Conflict\( (Y, X) \), either \( r \) is discarded or
2. \( \exists c_i \in C(r) \) s.t. \( +\partial_X c_i \in P(1..n) \), \( \forall c_j \in C(r), j < i, -\partial_X c_j \in P(1..n) \), and
3. \( \exists c_k, k \geq i \) s.t.
   1. \( 2.1.1' \) if \( k = i \) then \( +\partial c_k \in P(1..n) \), or
   2. \( 2.1.2' \) if \( k \neq i \) then either \( +\partial c_k \) and \( +\partial_D c_k \in P(1..n) \), or
      \( +\partial c_k \) and \( +\partial_X c_k \in P(1..n) \).

First, the agent chooses her level of commitment, that is the mode \( X \) among \( G, I, \) or \( SI \) to comply with (notice that in some cases this process is not particularly meaningful, e.g. desires, and this is the reason we decided to exclude a priori desires as a valid mental attitude from compliance checking). Then, we select the first element proved with modality \( X \) in the consequent of any applicable rule (element \( c_i \) in the proof condition (2.1)). We propose two variants of outcome compliance corresponding to sub-conditions (2.2.1’) and (2.2.1’’).

In the first case, we are compliant iff either \( c_i \) is proved as a belief (being the first element in the chain proved with modality \( X \)), or if there exists a following element \( c_k \) which has been proved as a desire as well as a belief. In the latter, we
are outcome compliant with respect to \( r \) if an element \( c_k \) following \( c_i \) has been proved as a belief, and its opposite has not been chosen as an outcome to achieve. It may be the case that, semantically but not syntactically, if \( \neg X \sim c_k \) is the case then \( +\partial_X c_k \), but this is left to further analysis.

Again, the counterparts \( +\partial_Y \bot \) and \( +\partial_X \bot, X \in \{G, I, SI\} \) are derived by strong negation applied to conditions for \( -\partial_Y \bot \) and \( -\partial_X \bot \), respectively.

We finally explain the last condition of (2.1.1.2'). In there, we allow a chain to be compliant even if a literal has been proven with outcome-compliant modality and has a factual derivation as well (\( +\partial c_k \) and \( +\partial_X c_k \in P(1..n) \)). This reflects the fact that a literal \( u_j \) following \( u_i \) in the chain of rule \( r \) can be proven as \( X \) due to another rule, we say rule \( s \), but nonetheless may positively affect the compliance of \( r \). To better understand, let us consider the following two rules where we state that both antecedents \( \Delta \) and \( \Gamma \) are the case.

**Example 7.**

\[
\begin{align*}
  r : \Delta &\Rightarrow \bigcup \ a \odot \ b \\
  s : \Gamma &\Rightarrow \bigcup \ c \odot \lnot a \odot d 
\end{align*}
\]

Suppose \( r > s \), and that the theory finally proves \( B b, B \sim a, \) and \( I \sim a \). Are we compliant with \( s \) given that \( -\partial_D \sim a \)? With the new condition we are and this is in line with our intuition: To be compliant we need to look at those elements which are, to some extent, desired or intended by the agent.

There and back again. Below we report the whole theory describing the Middle-Earth scenario formalised to show how compliance definitions work within our theoretical framework. For the sake of simplicity, we take the stance that whenever Frodo opts for a plan, all the actions to perform are either derivable from the theory, or considered as additional facts.

**Example 8.**

\[
\begin{align*}
  F &= \{\text{destroy}_\text{Ring}, \text{Gandalf}_\text{wise}, \text{Ring}_\text{bearer}, \text{Winter}\}, \\
  R &= \{r_1 : \text{Ring}_\text{bearer}, \text{destroy}_\text{Ring} \Rightarrow \bigcup \text{climb}_\text{MistyMts} \odot \text{Gap}_\text{Rohan} \odot \text{pass}_\text{Moria}, \\
                 &\quad r_2 : \text{severe}_\text{weather} \Rightarrow \lnot\text{climb}_\text{MistyMts}, \\
                 &\quad r_3 : \text{Winter} \Rightarrow \bigcup \lnot\text{climb}_\text{MistyMts} \odot \text{elf}_\text{scout} \\
                 &\quad r_4 : \text{Gandalf}_\text{wise} \Rightarrow \bigcirc \lnot\text{Gap}_\text{Rohan} \\
                 &\quad r_5 : \text{give}_\text{Ring}_\text{Gandalf} \Rightarrow \lnot\text{Ring}_\text{bearer} \\
                 &\quad r_6 : \text{pass}_\text{Moria} \Rightarrow \text{face}_\text{orcs} \\
                 &\quad r_7 : \text{other}_\text{ways}_\text{impracticable} \Rightarrow \text{pass}_\text{Moria} 
\end{align*}
\]
4.4. Running examples - Part I

We propose two running examples. Our aim being to explain all different aspects of our framework within a setting describing two business process-like scenarios. Each one of them starts with a description and the goal here is to show how an automated tool can transform a natural language description into a modal logic. These examples will be reprised in Section 6.3 where the focus will be on how to transform the logics associated into process graphs.

Example 9. PeoplEyes is an eyeglasses manufacturer. Naturally, their final goal is to produce cool and perfectly assembled eyeglasses. The final steps of the production process are to shape the lenses to glasses, and mount them on the frames. To shape the lenses, PeoplEyes uses a very innovative and expensive laser machine, while for the final mounting phase two different machines can be used. Although both machines work well, the first and newer one is more precise and faster than the other one, thus PeoplEyes prefers using the first machine as much as possible. Unfortunately, a new norm comes in force stating that no laser technology can be used, unless human staff wears laser-protective goggles.

If PeoplEyes has both human resources and raw material, and the three machines are fully working, but it has not yet bought any laser-protective goggles, all its goals would be achieved but it would fail to comply with the regulatory, since the norm for the no-usage of laser technology is violated and not compensated.

If PeoplEyes buys the laser-protective goggles, their entire production process also becomes norm compliant. If, at some time, the more precise mounting machine breaks, but the second one is still working, PeoplEyes still remains outcome...
compliant since the usage of the second machine leads to a state of the world where the objective of mounting the glasses on the frames is reached. Again, if PeoplEyes has no protective laser goggles and both the mounting machines are out of order, PeoplEyes’ production process is neither norm, nor outcome compliant.

Follows the formalisation into our logic.

\[
F = \{\text{lenses, frames, new\_safety\_regulation}\}
\]

\[
R = \{r_1 : \Rightarrow \text{eye\_Glasses}
\]

\[
r_2 : \Rightarrow \text{laser}
\]

\[
r_3 : \text{lenses, laser} \Rightarrow \text{glasses}
\]

\[
r_4 : \Rightarrow \text{mounting\_machine1}
\]

\[
r_5 : \Rightarrow \text{mounting\_machine2}
\]

\[
r_6 : \text{mounting\_mach1} \Rightarrow \neg \text{mounting\_machine2}
\]

\[
r_7 : \text{frames, glasses, mounting\_machine1} \Rightarrow \text{eye\_Glasses}
\]

\[
r_8 : \text{frames, glasses, mounting\_machine2} \Rightarrow \text{eye\_Glasses}
\]

\[
r_9 : \text{new\_safety\_regulation} \Rightarrow \neg \text{laser} \& \text{goggles}
\]

\[
r_{10} : \Rightarrow \text{mounting\_machine1} \& \neg \text{mounting\_machine2}
\]

\[>^{sm} = \{r_6 > r_5\}.
\]

We assume PeoplEyes has enough resources to start the process by setting lenses and frames as facts. Rule \(r_1\) states that producing eye\_Glasses is the main objective (+\(\partial\text{eye\_Glasses}\), we choose intention as the mental attitude to comply with); rules \(r_2\), \(r_4\) and \(r_5\) describe that we can use, respectively, the laser and the two mounting machineries. Rule \(r_3\) is to represent that if we have lenses and a laser machinery available, then we can shape glasses; in the same way, rules \(r_7\) and \(r_8\) describe that whenever we have glasses and one of the mounting machinery is available, then we obtain the final product. Thus the positive extension for belief +\(\partial\) contains laser, glasses, mounting\_machine1 and eye\_Glasses. In that occasion, rule \(r_6\) along with \(>\) prevent the using of both machineries at the same time and thus –\(\partial\text{mounting\_machine2}\) (we assumed, for illustrative purpose even if unrealistically, that a parallel execution is not possible). When a new safety regulation comes in force (\(r_9\)), the usage of the laser machinery is forbidden, unless protective goggles are worn (+\(\partial\neg\text{laser}\) and +\(\partial\neg\text{goggles}\)). Finally, rule \(r_{10}\) is to describe the preference of using mounting\_machine1 instead of mounting\_machine2 (hence we have +\(\partial\neg\text{mounting\_machine1}\) and –\(\partial\neg\text{mounting\_machine2}\)).
Since there exists no rule for goggles, the theory is outcome compliant, but not norm compliant. If we add goggles to the facts and we substitute $r_2$ with

$$r'_2 : \text{goggles} \Rightarrow \text{laser}$$

then we are both norm and outcome compliant, as well as if we add

$$r_{11} : \text{mounting}_1 \_ \text{broken} \Rightarrow \lnot \text{mounting}_1$$

to $R$ and $\text{mounting}_1 \_ \text{broken}$ to $F$. Notice that, with respect to laser, we are intention compliant but not social intention compliant (given O-lenses). This is a key characteristic of our logic: The system is informed that the process is compliant but some violations have occurred.

The second example is to explain how the same preconditions may occur in multiple rules; its utility will be clear in Section 6.3 to show some graph simplification patterns proper of the algorithms proposed in that chapter.

**Example 10.** A new Italian pizzeria has recently opened in Brisbane, Governatore’s Pizza. They offer an Italian, traditional menu, with only five types of pizza:

- **Margherita:** mozzarella, tomato sauce;
- **Spring:** mozzarella, cherry tomatoes;
- **Devil:** mozzarella, tomato sauce, spicy sausage, kalamata olives;
- **Autumn:** mozzarella, mushrooms, potatoes, bacon;
- **Marianna:** mozzarella, basil pesto, mushrooms.

Naturally, the clientele is well aware that each pizza is made with real Italian dough (hence dough is not reported in the previous menu but is present in the following list of recipes).

$$F = \{\text{dough, mozzarella, tomato_sauce, cherry_tomatoes, sausage, olives, mushrooms, potatoes, bacon, pesto, Imargherita, Ispring, Idevil Iautumn, Imarianna}\},$$

$$R = \{r_1 : \text{dough, mozzarella, tomato_sauce} \Rightarrow \text{margherita}$$

$$r_2 : \text{dough, mozzarella, cherry_tomatoes} \Rightarrow \text{spring}$$

$$r_3 : \text{dough, mozzarella, tomato_sauce, sausage, olives} \Rightarrow \text{devil}$$

$$r_4 : \text{dough, mozzarella, mushrooms, potatoes, bacon} \Rightarrow \text{autumn}$$

$$r_5 : \text{dough, mozzarella, pesto, mushrooms} \Rightarrow \text{marianna}\},$$

$$> = \emptyset.$$

As before, we assume that the pizzeria has sufficient ingredients to make as
many pizzas ordered. Rules in the theory describe recipes to prepare terrific pizzas. As stated before, this running example is not particularly interesting from the point of view of the corresponding logical theory, but it will be essential in Chapter 6 to illustrate how the synthesis algorithms are able to simplify the structure of a given process graph by recognising parallel and choice patterns.

4.5. Theory extension calculus: Summary and related work

This chapter provided a new proposal for extending Defeasible Logic to model cognitive agents interacting with obligations. These efforts were necessary in order to bind notions of norm and outcome compliances within a framework able to interrelate different facets of knowledge (beliefs, normative constraints, and mental attitudes) with one another.

Concerning outcome compliance, we chose that the agent should strive either for an element derived as desire, or derived as the selected outcome compliance modality. Another possibility to be compliant would have been to consider an element just with factual derivation and without considering if such an element would have been proved as an outcome. This choice is justified by the fact that, even without a derivation, each of the alternatives in a chain were given by the agent and may not been look at as forced by external factors. We believe that this solution is not satisfactory since the agent would be forced to pursue something neither desired, nor intended.

We proposed a fresh characterisation for the concepts of desires, goals, intentions, and social intentions which are motivational states obtained through a deliberative process based on various types of preferences among desired outcomes. In this sense, this contribution has strong connections with [Dastani et al., 2005a, Governatori and Rotolo, 2008b, Governatori et al., 2009b] but it completely rebuilds the logical treatment of agents’ motivational attitudes by presenting significant innovations in at least two respects.

First, while in [Dastani et al., 2005a, Governatori and Rotolo, 2008b, Governatori et al., 2009b] the agent deliberation is simply the result of the derivation of mental states from precisely the corresponding rules of the logic—besides conversions, intentions are derived using only intention rules, goals using goal rules, etc.—here, the proof theory is much more aligned with the BDI intuition, according to which intentions and goals are the results of the manipulation of desires. The conceptual result of the current logical formalisation is that this
idea can be entirely encoded within a logical language and a proof theory, by exploiting the different interaction patterns between the basic mental states, as well as the derived ones. In this perspective, our framework is significantly more rich than the one in BOID [Broersen et al., 2002], which uses different rules to derive the corresponding mental states and proposes simple criteria to solve conflicts between rule types.

Second, the framework proposes a rich language expressing two orthogonal concepts of preference among motivational attitudes. One is encoded within $\odot$ sequences, which state (reparative) orders among homogeneous mental states or motivations, and which are contextual. The second type of preference is encoded via the superiority relation between rules: the superiority can work locally between single rules of the same or different types, or can work systematically by stating via the $\text{Conflict}(X, Y)$ relation that two different motivations $X$ and $Y$ collide, and $X$ always overrides $Y$. The interplay between these two preference mechanisms can help us in isolating different and complex ways for deriving mental states, but the resulting logical machinery is still computationally tractable, as the algorithmic analysis in the present chapter has proved.

Finally, since the preferences allow us to determine what preferred outcomes are adopted by an agent (in a specific scenario) when previous elements in $\odot$-sequences are not (or no longer) feasible, our logic in fact provides an abstract semantics for several types of goal and intention reconsideration.

The fact that most BDI agent programming languages only deal with procedural aspects of goals shows the existing gap between BDI theory and implementation. Typically, BDI agents repeatedly revise their beliefs to accommodate newly perceived information and reason and deliberate on the basis of their beliefs, desires and existing intentions, about what actions are to be taken next and how to modify their intentions as time goes by and actions are performed (this approach has its roots in [Bratman, 1987, Dennet, 1987]). Intention reconsideration was expected to play a crucial role in the BDI paradigm [Bratman, 1987, Cohen and Levesque, 1990] since intentions obey the law of inertia and resist retraction or revision, but they can be reconsidered when new relevant information comes in [Bratman, 1987]. Despite that, the problem of revising intentions in BDI frameworks has received little attention. A very sophisticated exception was by van der Hoek et al. [2007], where revisiting intentions mainly depends on the dynamics of beliefs but the process is incorporated in a very complex framework for reasoning about mental states. Recently, Shapiro et al. [2012] discussed how to revise the commitments to planned activities because of mutually conflicting
intentions, which interestingly has connections with our work. How to employ our logic to give a semantics for intention reconsideration is not the main goal of this dissertation (see below for more detailed comments on [Shapiro et al., 2012]).

At a more general perspective, an annotated discussion of related work is presented below.

Our framework shares the motivation with [Winikoff et al., 2002], where the authors provide a logic to describe both the declarative and procedural nature of goals. The nature of the two approaches lead to conceptually different solutions. For example, they require a goal, as in [Hindriks et al., 2000], “not to be entailed by beliefs, i.e., that they be unachieved”, while our beliefs can be seen as ways to achieve goals. Other requirements as persistence, or dropping a goal when reached, cannot be taken into account. Consider their example about the cat who wants to get on the table to eat the food. The reconsideration of its plan to jump on the shelf and fortuitously find the food must be considered within our framework as a change of the facts describing the initial scenario, which triggers a new run of the algorithms to compute the extension of the new theory. By the nature of the example, the goal to eat the food is still achievable (by jumping on the shelf); the difference is that now the goal of eating the food is reached by following different rules in the theory. In the mentioned framework the cat has the goal to jump on the table as a consequence of wanting to eat, while we state that the goal $G_{food}$ may be achieved by the rule $r: table \Rightarrow B_{food}$. Winikoff et al. [2002] admit, as we do for desires, that it is possible to have inconsistent goals, i.e., $Gp$ and $G\neg p$. They motivate by saying that they both can be reached at different times, while we allow opposite desires, $Dp$ and $D\neg p$, but not for the other mental attitudes.

Hindriks and van Riemsdijk [2007, 2008] propose a layered rational action selection architecture which specifies a rational mechanism to choose actions in which hard and soft constraints are integrated. The introduction of maintenance goals provides tools to restrict the options for action selection, while achievement goals are seen as hard constraints which must be satisfied, in contrast with soft constraints in the form of a set of preferences allowing the agent to distinguish preferred courses of action. Apart from the known complexity problem of using a Linear Temporal Logic (LTL) formalisation, we found two main drawbacks in their work, the progression operator and the ability of lookahead. The progression operator is not complete; the progression is the transformation of a formula into a new one where the parts of the formula which have already been achieved are
set to true. Thus, it cannot be detected whether an achievement goal (notice, a hard constraint) may never be achieved. Then, an agent with a finite lookahead (which is the standard) is not able to conclude that an action will prohibit the realisation of a given goal. More interesting for future works is their Layer 3, where the agent uses the soft constraints to select those traces that maximise an utility function. Even here, the adopted notation is hard to grasp. To represent the statement “the agent prefers to go first to location $A$ before going to location $B$” they use the notation $\neg (\neg \text{at}(A) \cup \text{at}(B)) \land \text{at}(B)$, which in our opinion is less intuitive than ours $\Rightarrow \text{at}_A \cup \text{at}_B$ and $\text{at}_A \Rightarrow \text{at}_B$. In addition, there are some concerns that temporal logics, and in particular LTL, might not be able to model normative requirements as shown by Governatori [2014].

In [Dignum et al., 2000, 2002], the authors propose a logical representation of agent mental and motivational attitudes, focusing on goal generation. By the authors’ own admission, the logic they develop is very convoluted: They align to the BDI literature in the interpretation of beliefs, goals and intentions, where goals and intentions are taken as primitive notions and thus not dependant on one another. In our framework, the only primitive notion among mental and motivational attitudes is that of outcome; even desires must pass through the agent’s “preference filter”, which allows to have different desires but which skims among agent’s inner preferences. Another difference is that they describe beliefs as statements of properties that an agent esteems to be true but that can possibly be false. We are aligned with their perspective that desires might be, in principle, impossible to achieve, whereas goals are required to be both individually and jointly achievable (replacing their goals with our intentions). They make distinction between norms and obligations (we do not): The formers are considered more stable and abstract, while obligations are a consequence of direct actions. Moreover, they use a similar filtering mechanism, obtaining goals by filtering desires through obligations, and related desires are ordered by the preference relation $>_D(i,w)$ on possible worlds.

Shapiro and Brewka [2007] deal with goal change by extending the framework of [Shapiro et al., 2007]. The authors take into account the case when an agent readopts goals that were previously believed to be impossible to achieve up to revision of her beliefs. They model goals through an accessibility relation over possible worlds. This is similar to our framework where different worlds are different assignments to the set of facts. Similarly to us, they prioritise goals as a preorder $\leq$: an agent adopts a new goal unless another incompatible goal prior in the ordering exists. This is in line with our framework where if we change the set
of facts, the algorithms will compute a new extension of the theory where two opposite literals can be proved as D (desire) but only one as I (intention). Notice also that the ordering used in their work is unique and fixed at design time, while in our framework chains of outcome rules are built through a context-dependent partial order which, in our opinion, models more realistic scenarios.

Dastani et al. [2006] present three types of declarative goals: Perform, achievement, and maintenance goals. In particular, they define planning rules which relate configurations of the world as seen by the agent (i.e., her beliefs). A planning rule is considered correct only if the plan associated to the rule itself allows the agent to reach a configuration where her goal is satisfied. This is strongly connected to our idea of belief rules, which define a path to follow in order to reach an agent outcome. Besides, as already stated in Section 3.1, this kind of research based on temporal aspects is orthogonal to ours. Analogously to our research, van Riemsdijk et al. [2008] as well as Dastani et al. [2011] propose a unifying framework specifying different facets of the concept of goal. However, several aspects make a comparative analysis between the two frameworks unfeasible. Indeed, their analysis is merely taxonomical, nor does it address how goals are used in agent logics, as we precisely do. Moreover, like in [Dastani et al., 2006], their taxonomy considers dimensions that are orthogonal to ours.

Dastani et al. [2008] present a simplified version of a programming language to implement systems where the behaviour of an agent is regulated by norm-based artefacts (norms being enforced by enforcement and regimentation). Via regimentation, “the system prevents an agent from performing a forbidden action”, while enforcement tells the actions the agent must perform after a norm violation in order to return to an acceptable state. The formal framework ISLANDER/AMELI [Esteva et al., 2004] has been proposed to specify norms in institutions; in that framework, unrealistically in our opinion, norms can never be violated. Instead, the framework MOISE+ [García-Camino et al., 2005, Hübner et al., 2007] allows violations but such a system is neither able to handle how to respond in case of violations, nor implements mechanisms for monitoring and sanctioning. In [Dastani et al., 2008], the authors represent norms as count-as rules, brute facts describe the environment along with external actions (which correspond to our initial set of facts plus beliefs rules), while institutional facts constitute the normative state. An operational semantics is given, which is sound but only weakly complete. By using count-as rules, de facto, the authors implement a system similar to ours, with rules of three modes (linking (i) Brute states to normative ones, (ii) Institutional facts to institutional facts, and (iii)
Normative states to brute states to model sanctions). Two main issues lie in that approach: They cannot express chains of violations-reparations and, most importantly, their method to verify whether the program implementing the normative artefact is sound works in PSPACE (given that it checks all possible interleavings of actions and the underlying logic is the Propositional Dynamic Logic).

van Riemsdijk et al. [2009a] share our aim to formalise goals in a logic-based representation of conflicting goals and propose two different semantics to represent conditional and unconditional goals. They model the former by exploiting default logic which has strong similarities with Defeasible Logic. Nonetheless, their central thesis, supported by Prakken [2006], is that only by adopting a credulous interpretation it is possible to have conflicting goals. However, we believe that a credulous interpretation is not suitable if an agent has to deliberate what her primary goals are in a particular situation at a particular time. On the other hand, a credulous interpretation is appropriate to determine what are the acceptable outcomes for a scenario, but then, in the deliberation phase the agent has to commit to one of the mutually incompatible options. In our framework, we opted for a skeptical interpretation of the concepts we call goals, intentions, and social intentions, while we adopt a credulous interpretation for desires. Moreover, they do not take into account the distinction between goals and related motivational attitudes (as in [van Riemsdijk et al., 2008, Dastani et al., 2011, 2006]). The characteristic property of intentions in these logics is that an agent is required to be committed to its intentions. That is, an agent may not drop intentions for arbitrary reasons, which means that intentions have a certain persistence. That being the case, their analysis results orthogonal to ours.

Meneguzzi and Luck [2009] provide a technique that enables agents to process norms and to adapt their behaviour at runtime in response to newly accepted norms so as to comply with them, should they choose to do so. In order to work run-time, a procedure needs to be computationally efficient, and thus their approach has a similar aim with respect to ours. Even though their notion of obligation as denoting that an agent “must act to accomplish something” is quite different than ours, their representation \( \text{norm}(\text{Activation}, \text{Expiration}, \text{Norm}) \) is very similar: \text{Activation} and \text{Expiration} stand for the antecedents of an obligation rule while \text{Norm} is the conclusion. We found some cons in their approach. First, there is an overburden of notation. Second, their monotonicity assumption leads the proposed algorithms to perform too many (and not strictly necessary) operations when a norm expires. Given that the authors’ focus is on
understanding what happens and what to do either when a new norm impacts on the plans of the agent, or when an existing norm expires, that work is orthogonal to ours and lies in the field of process revision under compliance. Finally, they do not consider the scheduling of plans with respect to possible sanctions. Nonetheless, their approach can be a useful reference point for future works aiming at extending our logic with explicit time.

Vasconcelos et al. [2009] propose mechanisms for the detection and resolution of normative conflicts. They resolve conflicts by manipulating the constraints associated to the norms’ variables, as through curtailment that is reducing the scope of the norm. In other works, we dealt with the same problems in defeasible deontic logic [Governatori et al., 2013a,c]. We found three problems in their solution: (i) The curtailing relationship is rather less intuitive than our preference relation $\succ$, (ii) Their approach seems too convoluted in solving exceptions (and they do not provide any mechanism to handle reparative chains of obligations), and (iii) The space complexity of their $\text{adoptNorm}$ algorithm is exponential.

With similar aims and in the same research area is the work of van Riemsdijk et al. [2009b], where the focus is on analysing which kind of reasoning an agent should be able to do when acting in an organisational environment, and how such agents might be programmed. Their research is mainly (i) A motivational work on why it is important for agents to be able to reason within an organisation, and (ii) An outline of possible application areas (such as human-agent teamwork, or open systems). As the authors focus on the role played by an agent as part of an organisation, that work is not strictly related to the current work of this Ph.D. thesis, but is instructive for possible lines of research on how (and if it is meaningful) to increment our logics in order to capture features of role specifications. One of the main challenges is to bridge the gap between the possibly abstract specification of the agent role and the concrete actions the agent has to take. Secondly, deciding on whether to violate some norms by weighing benefits of breaking the rules against possible negative consequences. This would let the system be able to deal with delicate situations as their firemen-police officers example, where an emergency situation has occurred and the firemen, the first duty forces arrived on the scene, need to decide whether to set a road block, preventing in losing lives but violating the norm under which only police officers are allowed to set road blocks, or to comply with that policy and risk more serious collateral damages.

Grant et al. [2010b], the authors present a formal model where a BDI agent
deliberates about her intentions taking into account the intentions of other agents. Here, the constraints given by other agents’ intentions is an antonomasia for our obligations. Notice that in their logics intentions correspond to plans chosen for execution, and the means an agent possesses to bring about certain states of affairs is captured by a set of recipes such as \( \langle \alpha, \varphi \rangle \), stating that the execution of action \( \alpha \) will accomplish a state of affairs satisfying \( \varphi \). Finally, they represent beliefs as well as desires as logical formulae, closed under deduction, while the agent can choose between and commit to possible set of intentions via the value and cost functions. Given that our model satisfies their condition of being a WRBDI structure (Weakly Rational BDI), we speculate that in future lines of research we shall be able to adapt their function \( \text{ben}_i(I, B) \), which defines the benefits agent \( i \) would obtain from the set of intentions \( I \), to our more efficient framework (in their framework, checking the consistency of intentions is co-NP). This will also be an interesting study on what they call intentions in equilibrium, that is agents adopting locally rational intentions, where an agent assumes that any benefit it derives from its own intentions alone, against Nash stabilities (simple, objective, and subjective), where the agent also uses other agents’ intentions in planning her own.

Grant et al. [2010a] take intentions as a set of recipes. The paper proposes AGM-style postulates for belief, intention, and desire revision. In [Governatori et al., 2012a], we have already shown how such postulates are inadequate for revising non-monotonic logics. We have seen in [Grant et al., 2010b] as the notion of recipe as the tuple \( \langle \alpha \theta \rangle \) is comparable to a belief rule. Something very interesting for future works is how they think of the cost function, which is not required to be additive; for example, the cost of performing two actions may be lower than the sum of performing them separately. Of less importance, instead, is the computational result claiming that the problem of checking if a set of recipes (plans) is sub-optimal is NP-complete. On the contrary, we find it very interesting how the selection between alternative sets of possible plans is achieved by doing a cost/value analysis, via the corresponding functions. That methodology gives us an idea on how to modify our algorithms, not just by implementing such functions, but in the more general compliance perspective. Once the computation of the logic extension is done, Algorithms 4 and 5 build up the process graph by adding, with a backward procedure, every single rule “in the positive extension”. If we associate to such rules the value and cost function, the algorithms can “add up” gains and costs of violating norms at each step. In this way, we do not have only a final metric for each path, but
also every partial evaluation (thus not affecting the overall complexity). If our intuition is sound, this will greatly improve the general computation of finding the more profitable paths, as well as give local information on where some obligations were violated and, consequently, help process revision methodologies. Finally, the computational efficiency of our problem helps us avoid their main source of difficulty: Change in beliefs, which should be followed by a parsimony requirement in changing the theory. As soon as some new beliefs are added to the set of facts, the run of our algorithms give the new set of objectives and norms in polynomial time; moreover, in [Governatori et al., 2012a] we showed how little significance the concept of minimal change has when dealing with non-monotonic reasoning (and in general with real life scenarios).

Shapiro et al. [2012] propose an orthogonal work with respect to ours given that the authors study how to revise a formal system, but given that they tackle both problems of conflicting intentions and preferences among sets of intentions, that paper is of interest to compare the conceptual ideas underpinning our and their approach. To begin with, we think that their notion of intention not as “something the agent would like to achieve” but as “the way the agent intend to achieve some goals” is not just misleading, but semantically wrong. An intention should represent a mental attitude, and be distinct from the notion of goal (an intention should not fail). In our perspective, intentions and plans are different entities: “what I want” is different from “how do I get it”. Their model is based on AgentSpeak languages, where an agent selects a plan from the system’s know-how information for execution. It is clear how such works lay on a conceptual different level with respect to ours, since our aim was to build what they consider such a know-how plan library (our business process model). Given our choice of Defeasible Logic as the agent-description language, we depart from the operational semantics they adopt [Rao and Georgeff, 1992]. Specifically, their plan library and action library are represented within our framework by means of belief rules. A sequence of actions can be seen as a derivation. Their transition relation, describing a single step execution resulting in a configuration of the environment, has the counterpart in our belief rules. Moreover, the language itself does not provide any default built-in failure handling, and provides a low-level commitment to intentions given that an intention is dropped as soon as it cannot evolve. Finally, the formalisation of our logical apparatus ensures both consistency and coherency with respect to beliefs, whilst Shapiro et al. [2012] force them within their system. We think our formalisation is as intuitive as theirs, and maybe even easier to understand, as we shall prove
in Chapter 6.2, leads to a computationally more efficient framework (against AgentSpeak languages which have a general complexity of PSPACE).

Shoham [2009] considers the concept of intention; the approach adopted is based on database perspective. Nonetheless it is most interesting for us from an ontological perspective given that the author thoroughly discusses the notion of intention as a mental state within logics of rational agency. He describes the first category of mental states as information attitudes which capture agents’ assessment of whether a certain fact holds true. But then Shoham states that “information attitudes constitute just one facet of mental state. In contrast, motivation attitudes capture something about the agents’ preference structure, and action attitudes capture his inclination towards taking certain actions. In a typical theory, the action attitudes mediate between the informational and motivational attitudes; the agent’s choice of action is dictated by his wants and beliefs. Into these two broad camps fall notions such as desire, goal, intention.” Here, connections with our framework are evident. Beliefs fall into the category of information attitudes, the preference order established by an outcome rule describes the agent’s motivational connotation, while action attitudes are captured by our Convert and Conflict relations, which lead the agent in choosing her desires, goals, (social) intentions. Even when, later on in his work, the author uses the example of the “morning star - evening star” (which is actually the same, Lucifer) to disqualify beliefs as the definition of knowledge, we think that our position in this Ph.D. dissertation is aligned with his, where the only difference is not in the essence but in the terminology we adopted to simplify the notation; our work did not want to be a philosophical dissertation about the notion of belief, or knowledge in general.

We end by reporting Shoham’s thoughts when discussing about knowledge and the meaning of the K axiom with respect to cryptography. Shoham states that one can come up with always new examples to disqualify a theory, but in the end “One should be explicit about the intended use of the theory, and within the scope of this intended use one should require that everyday intuition about the natural concepts be a useful guide in thinking about their formal counterparts”. In this respect, we think we succeeded in our endeavour.
We shall present procedures and algorithms apt to compute the \textit{extension} of a finite, modal defeasible theory (Section 5.2), in order to provide a bound for the computational complexity of the logic previously introduced. The computation of the logical extension is necessary to pinpoint which elements of the initial declarative specifications will be part of the logical derivations and as such contribute to the compliances of the agent. Those particular derivations will later be translated into the graph describing the process.

Starting point of the algorithms of this chapter being the ideas proposed in [Maher, 2001, Lam and Governatori, 2011] but a consistent evolution has been proposed in how to manage chains of outcomes and how those can be modified with the information obtained by belief and obligation literals.

This chapter is based on [Governatori et al., 2013b].

\textbf{Outline}

The present chapter is structured as follows. We shall start by introducing in Section 5.1 the notation the algorithms adopt; there we shall present two important operations used by the algorithms to shorten (simplify) chains, namely \textit{truncation} and \textit{removal}. The main part of the chapter being Section 5.2 which, besides reporting the algorithms, details descriptions of their behaviour. Section 5.3 ends by showing the computational results. To facilitate reading, we decided to postpone the technicalities of proving Lemmas 35–48 and Proposition 34 (concerning the transformations involved within the algorithms) to the later
5.1. Notation for the algorithms

For the sake of clarity, from now on ◯ denotes a generic modality in MOD, ◊ a generic modality in MOD \ {B}, and □ a fixed modality chosen in ◯. Moreover, whenever □ = B, we shall treat literals □l and l as synonyms. To accommodate the Convert relation to the algorithms, we recall that $R^B ▷$ denotes the set of belief rules which can be used for a conversion to modality ◊. The antecedent of all such rules: (i) Is not empty, (ii) It does not contain any modal literal. Furthermore, for each literal $l$, $l ◯$ is the (initially empty) set such that $± □ l 2 □ ◯ l$. Given a modal defeasible theory $D$, a set of rules $R$, and a rule $r 2 R ▷ l$, we expand the superiority relation $>$ by incorporating the Conflict relation into it; this will facilitate the computation. According to these statements we define:

1. $r_{sup} = \{s 2 R : (s, r) \in >\}$ and $r_{inf} = \{s 2 R : (r, s) \in >\}$ for any $r 2 R$;
2. $HB_D$ as the set of literals such that the literal or its complement appears in $D$, where ‘appears’ means that it is a sub-formula of a modal literal occurring in $D$;
3. The modal Herbrand Base of $D$ as $HB_D = \{l | □ l 2 MOD, l 2 HB_D\}$.

Hence, the extension of a defeasible theory is defined as follows.

**Definition 19.** Given a modal defeasible theory $D$, the defeasible extension of $D$ is defined as

$$E(D) = (+\partial_\Box, -\partial_\Box),$$

where $\pm \partial_\Box = \{l 2 HB_D : D + \pm \partial_\Box l\}$ with $\Box 2 MOD$.

We state that two defeasible theories $D$ and $D'$ are equivalent whenever they have the same extensions, i.e., $E(D) = E(D')$.

The next definition introduces two syntactical operations on the consequent of rules used by the algorithms. They represent the two fundamental mechanisms we shall use in the algorithms to simplify chains of reparative literals in the head of a rule.

**Definition 20.** Let $c_1 = a_1 \odot \ldots \odot a_{i-1}$ and $c_2 = a_{i+1} \odot \ldots \odot a_n$ be two (possibly empty) ◮-expressions such that $a_i$ does not occur in either of them, and $c = c_1 \odot a_i \odot c_2$ is an ◮-expression. Let $r$ be a rule with form $A(r) \Rightarrow_X c$. We define the operation of truncation of the consequent $c$ at $a_i$ as:

$$A(r) \Rightarrow_X c!a_i = A(r) \Rightarrow_X c_1 \odot a_i.$$
5.2. ALGORITHMS FOR THE LOGICAL EXTENSION CALCULUS

We define the operation of removal of \( a_i \) from the consequent \( c \) as:

\[
A(r) \Rightarrow c \odot a_i = A(r) \Rightarrow c_1 \odot c_2.
\]

Section 4.1 defined the language of our modal theory. In there, each rule is defined by a unique name \( r \in \text{Lab} \). Operations of removal and truncation change consequents by acting on the chains of a given rule. To capture outcome compliance, we need to bind modified rules with the “original” ones.

**Definition 21** (Rule label retrieval). Given a modal defeasible theory \( D = (F, R, >) \), define the bijective function \( \text{label} : R \rightarrow \text{Lab} \) such that

\[
\text{label}(r : A(r) \leftrightarrow C(r)) = r.
\]

Furthermore, notice that removal may lead to rules with an empty consequent which therefore would no longer make them rules according to the definition of the language. However, in the description of the algorithms we allow for such expressions but then such rules will not be in any \( R[q, i] \) for any \( q \) and \( i \). Thus, in such cases, the operation effectively disables the rule.

Given \( \Box \in \text{MOD} \), the sets \( +\hat{\Box} \) and \( -\hat{\Box} \) denote, respectively, the global sets of positive and negative defeasible conclusions (i.e., the set of literals for which condition \( +\hat{\Box} \) or \( -\hat{\Box} \) holds), while \( \hat{\Box}^+ \) and \( \hat{\Box}^- \) are the corresponding temporary sets, that is the sets computed at each iteration of the main algorithm. Moreover, to simplify the computation we do not operate on outcome rules: for each rule \( r \in R^U \) we create instead a new rule for desire, goal, intention, and social intention (respectively, \( r^D \), \( r^G \), \( r^I \), and \( r^{SI} \)). Accordingly, for the sake of simplicity, in the rest of this dissertation we shall use the expression “the intention rule” as a shortcut for “the clone of the outcome rule used to derive intentions”.

5.2. Algorithms for the logical extension calculus

The idea of all algorithms is to use the operations of truncation and elimination to obtain, step after step, a simpler but equivalent theory. In fact, proving a literal does not give just local information about the element itself, but reveals which rules will be discarded, or reduced in their head or body.

**Observation 22.** Let us assume that, at a given step, the algorithm proves literal \( l \). At the next step,

1. the applicability of any rule \( r \) with \( l \) in its antecedent \( A(r) \) does not depend
on $l$ any longer. Thus, we can safely remove $l$ from $A(r)$.

2. Any rule $s$ where $\bar{l}$ is in its antecedent $A(s)$ is discarded. Consequently, any superiority tuple involving this rule is now useless and can be removed from the superiority relation as well.

3. We can shorten chains by exploiting conditions of Definitions 7 and 8. For example, if $l = Om$, we can truncate chains for obligation rules at $\sim m$ and eliminate it as well.

Algorithm 1 DefeasibleExtension is the core algorithm to compute the extension of a defeasible theory. The first part of the algorithm (Lines 1–7) sets up the data structure needed for the computation. Lines 8–11 are to handle facts as immediately provable literals.

The main idea of the algorithm is to check whether there are rules whose body is empty: Such rules are trivially applicable and they can produce conclusions with the right mode. However, before asserting that the first element for the appropriate modality of the conclusion is provable, we need to check whether there are rules for the complement (again with the appropriate mode): In that case such rules must be weaker than the applicable rules. The information about which rules are weaker than the applicable ones is stored in the support set $R_{\text{inf}}$. The idea of using $R_{\text{inf}}$ was first proposed by Lam and Governatori [2011], but we considerably revised and improved it to suitably adapt $R_{\text{inf}}$ to our more complicated Convert and Conflict relations. If a literal is evaluated to be provable (with the appropriate mode) the algorithm calls Algorithm 2 Proved; if a literal is rejected then the Algorithm 3 Refuted is invoked. These two procedures do the appropriate bookkeeping by applying transformations to reduce the complexity of the theory.

We now proceed into a more detailed description of the algorithm, which starts by initialising the global and temporary sets of defeasible conclusions to the empty sets for all modalities (Line 1).

For every outcome rule, the algorithm makes a copy of the same rule for each mode corresponding to a goal-like attitude (Line 2). The outcome rule set is erased from $R$ at Line 3. At Line 6, the algorithm creates a support set to handle conversions from a belief rule through a different mode. Consequently, the superiority relation is updated at Line 7 to handle these new rules since a superiority between two belief rules is inherited by the new $\Diamond$ rules we just created through the Convert relation at Line 6. Notice that we also augment the superiority relation by incorporating the rules involved in the Conflict relation as well.
Algorithm 1 DefeasibleExtension (Part 1)

Input: A defeasible theory $D$ and a mode $X \in \{G, I, SI\}$.

Output: The (positive) extension $E(D)$ of $D$ if $D$ compliant, the error message

The theory is not compliant otherwise.

1. $+\mathbb{1};, -\mathbb{1} \leftarrow \emptyset$; $-\mathbb{1}; \leftarrow \emptyset$
2. $R \leftarrow R \cup \{r^\square : A(r) \Rightarrow C(r) | r \in R^U\}$, with $\square \in \{D, G, I, SI\}$
3. $R \leftarrow R \setminus R^U$
4. $nCObl \leftarrow R^O \cup R^{B,O}$
5. $nCOut \leftarrow R^X \cup R^{B,X}$
6. $R^{B,O} \leftarrow \{r^\diamond : A(r) \Rightarrow C(r) | r \in R^B, A(r) \neq \emptyset, A(r) \subseteq \text{Lit}\}$
7. $\Rightarrow \cup \{(r^\diamond, s^\diamond) | r^\diamond, s^\diamond \in R^{B,O}, r > s \} \cup \{(r, s) | r \in R^\bullet \cup R^{B,O}, s \in R^O \cup R^{B,O}, \text{Conflict}(\mathbb{1}, \diamond)\}$
8. for $l \in F$
9. \hspace{1em} if $l = \square m$ then $\text{PROVED}(m, \square)$
10. \hspace{1em} if $l = \neg \square m \land \square \neq D$ then $\text{REFUTED}(m, \square)$
11. end for
12. $+\mathbb{1} \leftarrow +\mathbb{1} \cup +\mathbb{1};$; $-\mathbb{1} \leftarrow -\mathbb{1} \cup -\mathbb{1};$
13. $\mathbb{1} \leftarrow \emptyset$
14. repeat
15. \hspace{1em} $+\mathbb{1}; \leftarrow \emptyset$; $-\mathbb{1}; \leftarrow \emptyset$
16. \hspace{1em} for $\square l \in HB$
17. \hspace{2em} if $R^\square[l] \cup R^{B,\square}[l] = \emptyset$ then $\text{REFUTED}(l, \square)$
18. \hspace{1em} end for
19. \hspace{1em} for $r \in R^\square \cup R^{B,\square}$
20. \hspace{2em} if $A(r) = \emptyset$ then
21. \hspace{3em} $r_{\text{inf}} \leftarrow \{s \in R|(r, s) \in \}$; $r_{\text{sup}} \leftarrow \{s \in R|(s, r) \in \}$
22. \hspace{3em} $R_{\text{inf}} \leftarrow R_{\text{inf}} \cup r_{\text{inf}}$
23. \hspace{3em} Let $l$ be the first literal of $C(r)$ in $HB$
24. \hspace{3em} if $r_{\text{sup}} = \emptyset$ then
25. \hspace{4em} if $\square = D$ then
26. \hspace{5em} $\text{PROVED}(l, D)$
27. \hspace{5em} else
28. \hspace{6em} $\text{REFUTED}(\neg l, \square)$
29. \hspace{4em} $\text{REFUTED}(\neg l, \diamond)$ for $\diamond$ s.t. Conflict$(\square, \diamond)$
30. \hspace{4em} if $R^\square[\neg l] \cup R^{B,\square}[\neg l] \cup R^\bullet[\neg l] \setminus R_{\text{inf}} \subseteq r_{\text{inf}}$, for $\mathbb{1}$ s.t. Conflict$(\mathbb{1}, \diamond)$ then
31. \hspace{5em} $\text{PROVED}(l, \square)$
32. \hspace{4em} end if
33. \hspace{4em} end if
34. \hspace{3em} end if
35. \hspace{1em} end if
36. \hspace{1em} end for
37. \hspace{1em} $+\mathbb{1}; \leftarrow +\mathbb{1}; \cup +\mathbb{1};$; $-\mathbb{1}; \leftarrow -\mathbb{1}; \cup -\mathbb{1};$
38. \hspace{1em} $+\mathbb{1}; \leftarrow +\mathbb{1}; \cup +\mathbb{1};$; $-\mathbb{1}; \leftarrow -\mathbb{1}; \cup -\mathbb{1};$
39. until $+\mathbb{1} = \emptyset$ and $-\mathbb{1} = \emptyset$

=> (Continue)
Algorithm 1 Defeasible Extension (Part 2)

40: \( nCObl \leftarrow \{ r \in nCObl \mid \exists s \in R^O \cup R^{B,O}. A(s) = \emptyset \text{ and } \text{label}(r) = \text{label}(s) \} \)

41: \( nCObl \leftarrow nCObl \setminus \{ s \in R^O \cup R^{B,O} \mid C(s) \neq \emptyset \text{ and } \exists r \in nCObl. \text{label}(r) = \text{label}(s) \} \setminus \{ r \in nCObl \mid (l, 1) \in C(r). l \in -\partial_O \} \)

42: for \( r \in nCOut \) do
   \( \rightarrow \) outcome-compliance wrt. \( X \)
   Let \((l, i) \in C(r) \) be such that \( l \in +\partial_X \) and \( \forall l'(j) \in C(r). j < i \) and \( l' \in +\partial_X \)
   43: \( C(r) \leftarrow C(r) \oplus c \) such that \( (c, j) \in C(r) \) and \( j < i \)
   44: \( C(r) \leftarrow C(r) \oplus c \) such that \( c \notin +\partial_D \cup +\partial_X \)
   45: end for

46: \( nCOut \leftarrow nCOut \setminus \{ r \in nCOut \mid \exists l \in C(r) \text{ and } l \in +\partial_B \} \)

47: if \( nCObl = \emptyset \land nCOut = \emptyset \) then
   48: \( \rightarrow \) return \( +\partial_X \)
   49: else
   50: \( \rightarrow \) return The theory is not compliant
   51: end if

52: end

The set of facts is now analysed (for cycle at Lines 8–11): since facts are immediately proved literals, Algorithm 2 Proved is invoked for positively proved modal literals (i.e., proved with \( +\partial_X \)), while Algorithm 3 Refuted for negatively proved ones (i.e., those proved with \( -\partial_X \)).

The last two initialisation steps are to store the results just computed in the corresponding global sets (Line 12), and to create the set containing the rules weaker than the applicable ones (Line 13).

The algorithm now enters the main loop (repeat/until routine at Lines 14–39), which terminates the execution once no more modifications on the extension are made, i.e., when sets \( +\partial_X \) and \( -\partial_X \) are empty at the end of the iteration.

The aim of the for loop at Lines 16–18 is to discard any modal literal in \( HB \) for which there are no rules that can prove it (either directly, or through conversion).

We now iterate on every rule that can fire, that is rules with empty body (loop for at Lines 19–36 and if condition at Line 20); we collect the weaker rules in the set \( R_{infd} \) (Line 22). Since a consequent can be an \( \ominus \)-expression, the literal we are interested in is the (actual) first element of the \( \ominus \)-expression (Line 23). If no rule stronger than the current one exists, then the complementary conclusion is refuted by condition (2.3) of Definition 12 (Line 28). Consequently, literal \( l \) is refutable in \( D \) for any modality conflicting with \( \Box \) as well (Line 29). Notice that this reasoning does not hold for desires given that \( Dl \) and \( D\neg l \) might be true at the same time. Therefore, when \( \Box = D \) and the guard at Line 24 is satisfied, the
5.2. ALGORITHMS FOR THE LOGICAL EXTENSION CALCULUS

Algorithm directly invokes Algorithm 2 Proved (Line 26) since condition (2.3) of Definition 9 holds.

The next step is to check whether there are rules for the complement literal of the same modality, or of a conflicting modality. The rules for the complement should not be defeated by an applicable rule, i.e., they should not be in $R_{infd}$. If all these rules are defeated by $r$ (Line 30), then conditions for deriving $+\partial_\square$ are satisfied, and Algorithm 2 Proved is invoked.

Last steps of the algorithms test norm and outcome compliance. We initialised support sets at Lines 4 and 5. Norm compliance is the state of affairs where, given a particular norm, either (1) Such a rule is not violated, or (2) It is compensated. Thus, if (1) then the rule does not fire and this guarantees compliance. Concerning (2), the compliance is ensured if: (i) The rule is applicable, and (ii) An element in its chain is proved as obligation as well as belief.

For (i), $A(r)$ must be empty (Line 40). Concerning (ii), given element $l$ in an obligation chain, the algorithms remove $l$ if, and only if, $l$ is an obligation but it is not proved as belief. Moreover, the algorithms truncate after $l$ whenever (a) $l$ is proved as a belief, or (b) $l$ is not proved as obligation (in this situation $l$ is also removed from the chain). Consequently, an applicable obligation rule is in one of these two mutual exclusive conditions: Either the rule has one element in the chain, or its consequent is empty. These two statements follow. First, an obligation rule with a single literal in the consequent is a fulfilled obligation. Second, an obligation rule with an empty consequent represents a non-compliant situation, unless the first element of the chain has been proved as $-\partial_\square$ (Line 41). If the set resulting from the operations at Lines 40 and 41 is not empty, then a non-compliant situation has occurred.

Concerning the outcome compliance, Definition 18 indicates that we need to consider only the rules which have (at least) one element in the consequent proved with modality $X$. Of each of such rules (loop for at Lines 42–46), we consider the first element $l$ of the chain proved as $X$ (Line 43): We erase all elements which were before $l$ in the original chain (Line 44) as well as the ones after $l$ which were neither proved as desire, nor with modality $X$ (Line 45). For every applicable rule\footnote{Notice that, as before, a rule is applicable if the algorithms have emptied the set of antecedents.} for outcome used to derive literals with modality $X$, at least one of these elements needs to be proved as a belief as well (Line 47).

Here, we acted on the original rules. This is the case since we need to reconstruct all the eligible elements of the rules involved in the outcome compliance. In fact, three considerations are in order. First, when the repeat/until routine
at Lines 14–39 ends, each rule \( r \in R^X \) with \( X \in \{G, I, SI\} \) has a consequent made by a single literal. Second, desire literals appearing before \( l \) do not play a role in the outcome compliance. Third, if \( X \neq G \) then we also need to integrate those literals present in the original chain and proved as (social) intentions. This is necessary given that \( +\partial_X l \) does not imply \( +\partial_D \). Consequently, a literal may be an (social) intention but not a desire.

The algorithm ends the execution either at Line 49, or at Line 51. In the former case, the theory is both norm and outcome compliant; thus, the algorithm returns the extension of the input theory. In the latter case, the algorithm prints the message ‘The theory is not compliant’ on the screen.

**Algorithm 2 Proved**

```
1: procedure Proved(l ∈ Lit, □ ∈ MOD)
2: \( \partial_O^r \leftarrow \partial_O^r \cup \{ l \}; \ [\ ] \leftarrow [\ ] \cup \{ +\square \} \)
3: \( HB \leftarrow HB \setminus \{ \square l \} \)
4: if □ ≠ D then Refuted(¬l, □)
5: if □ = B then Refuted(¬l, I)
6: if □ ∈ {B, O} then Refuted(¬l, SI)
7: \( R \leftarrow \{ r : A(r) \setminus \{ \square l, \neg \square l \} \rightarrow C(r)\} \) \( r \in R, A(r) \cap \square l = \emptyset \)
8: \( R_B^O \leftarrow \{ r : A(r) \setminus \{ l \} \rightarrow C(r)\} \) \( r \in R_B^O, \neg l \notin A(r) \)
9: \( \leftrightarrow \ \\{ (r, s), (s, r) \in > | A(r) \cap \square l = \emptyset \} \)
10: switch (□)
11: \( \text{case B:} \)
12: \( R^X \leftarrow \{ A(r) \rightarrow_X C(r)\} \) \( | r \in R^X[\overline{l}, n] \} \) with \( X \in \{O, I\} \)
13: if +O ∈ ~\[\] then \( R^O \leftarrow \{ A(r) \rightarrow O C(r) \oplus \neg l | r \in R^O[\neg l, n] \} \)
14: if -O ∈ ~\[\] then \( R^SI \leftarrow \{ A(r) \rightarrow SI C(r)\} \) \( | r \in R^SI[\overline{l}, n] \} \)
15: \( \text{case O:} \)
16: if -B ∈ \[\] then \( R^O \leftarrow \{ A(r) \rightarrow O C(r) \oplus l | r \in R^O[l, n] \} \)
17: if -B ∈ ~\[\] then \( R^SI \leftarrow \{ A(r) \rightarrow SI C(r)\} \) \( | r \in R^SI[\overline{l}, n] \} \)
18: \( \text{case D:} \)
19: if +D ∈ ~\[\] then \( R^G \leftarrow \{ A(r) \rightarrow G C(r)\} \) \( | r \in R^G[\overline{l}, n] \} \)
20: \( R^G \leftarrow \{ A(r) \rightarrow G C(r)\} \) \( \neg l \oplus \neg l | r \in R^G[l, n] \} \)
21: end if
22: otherwise: \( R^D \leftarrow \{ A(r) \rightarrow D C(r)\} \) \( | r \in R^D[l, n] \} \)
23: end switch
24: end procedure
```

Algorithm 2 Proved is invoked when literal \( l \) is proved with modality \( □ \): this modality is the key to which simplifications on the rules can be done.

The computation starts by updating the relative positive extension set for modality \( □ \) and, symmetrically, the local information on literal \( l \) (Line 2); \( l \)
is then removed from $HB$ at Line 3. Proposition 16 parts 1–3 identifies the modalities literal $\neg l$ is refuted with when $\Box l$ is proved (if guards from Line 4 to 6). Lines 7 to 9 modify the superiority relation and the sets of rules $R$ and $R^{B,\Box}$ according to the intuitions given in Observation 22.

Depending on the modality $\Box$ of $l$, we have to perform specific operations on the chains (condition switch at Lines 10–25). A detailed description of each case follows according to the principles stated by conditions of Definitions 7 and 8.

**case B** at Lines 11–14, where $l$ has been proved as a belief. Proving a literal as a belief provides information related to obligation, intention, and social intention literals. First, conditions of Definitions 8 and 12 ensure that $\neg l$ may be neither an intention, nor a social intention. Thus, Algorithm 3 Refuted is invoked at Lines 5 and 6 which, in turn, removes $\neg l$ from every chain of intention and social intention rules (resp. at Line 18). Second, chains of obligation (resp. intention) rules can be truncated at $l$ since condition (2.1.2) (resp. condition (4.1.2)) of Definition 8 discards such rules for all elements following $l$ in the chain (Line 12). Third, if $+\partial_O \neg l$ has already been proved, then we remove $\neg l$ in chains of obligation rules since it represents a violated obligation. Fourth, if $-\partial_O \neg l$ is the case, then each element after $l$ cannot be proved as a social intention (if conditions at Lines 13 and 14, respectively). Accordingly, we truncate chains of social intention rules at $l$.

**case O** at Lines 15–17, where $l$ has been proved as an obligation. Proving a literal as an obligation provides information related to obligation and social intention literals. First, conditions of Definitions 8 and 12 ensure that $\neg l$ may be neither an obligation, nor a social intention. Thus, Algorithm 3 Refuted is invoked at Lines 4 and 6 which, in turn, i. Removes $\neg l$ from every chain of obligation and truncates such chains (Line 13), and ii. Truncates chains of social intentions at $\neg l$ if $-\partial l$ has already been proved (Line 14). Second, at Line 16, $l$ is removed from chains of obligation if it was proved as a belief as well. Finally, chains of social intentions are truncated at $l$ if $-\partial -$ has been proved (Line 17).

**case D** at Lines 18–22, where $l$ has been proved as a desire. If $\neg l$ has been proved as a desire as well (if guard at Line 19), then neither $l$, nor $\neg l$, nor any element following them may be goals. Therefore, at Lines 20–21, such chains are truncated while $l$ or $\neg l$ are removed.

**otherwise** at Lines 23–24, where $l$ has been proved either as a goal, or as an intention, or as a social intention. First, following conditions of Definitions 8 and 12, Algorithm 3 Refuted is invoked at Lines 4. For these cases, see the detailed description reported below. Line 24 truncates chains of the right mode (either G, I, or SI) at $l$. 

---

Ph.D. Thesis - 2014 Francesco Olivieri
Algorithm 3 Refuted

1:  procedure Refuted(l ∈ Lit, □ ∈ MOD)
2:  δ̂ −→ δ −→ |l|; l −→ l −→ {¬□}
3:  HB −→ HB \ [#□]
4:  R ← {r : A(r) \ (#□l) ↭ C(r) | r ∈ R, □l ∉ A(r)}
5:  R := R \ {r ∈ R : l ∈ A(r)}
6:  >−−⇒ ((r, s), (s, r) ∈ ⇒ □l ∈ A(r)}
7:  switch (□)
8:  case B:
9:     if +O ∈ l then R := {A(r) ⇒ O C(r) ⊂ l | r ∈ R[□][l, n]}
10:    if −O ∈ l then R := {A(r) ⇒ I C(r) | ¬l | r ∈ R[□][l, n]}
11: case O:
12:    R := {A(r) ⇒ O C(r) ⊂ l | r ∈ R[□][l, n]}
13:    if −B ∈ l then R := {A(r) ⇒ I C(r) | ¬l | r ∈ R[□][l, n]}
14: case D:
15:    R := {A(r) ⇒ X C(r) ⊂ l | r ∈ R[□][l, n]} with X ∈ {D, G}
16:    otherwise:
17:        R := {A(r) ⇒ □ C(r) ⊂ l | r ∈ R[□][l, n]}
18:  end switch
19:  end procedure

Algorithm 3 Refuted performs all necessary operations in the case where literal l is refuted with modality □.

The initialisation steps at Lines 2–6 follow the same schema exploited at Lines 2–9 of Algorithm 2 Proved. Again, the operations to be performed on chains vary according to the current mode □ (switch at Lines 7–19).

case B at Lines 8–11, where l has been refuted as a belief. Thus, condition (4.1.2) for ±δl of Definition 8 is satisfied for any literal after ¬l in chains for intentions. Accordingly, such chains can be truncated at ¬l. Furthermore, if +δOl has already been proved, then the obligation of l has been violated. Thus, l can be removed from all chains for obligations (Line 10). If instead −δOl holds, then the elements after ¬l in chains for social intentions satisfy condition (4.1.2) of Definition 8, and the algorithm removes them (Line 11).

case O at Lines 12–14, where l has been refuted as an obligation. First, an element not proved as an obligation does not need to be further compensated. According to condition (2.1.2) of Definition 8, chains for obligation containing l are truncated and the element itself is removed (Line 12). Symmetrically to the previous case B-Line 11, chains for social intentions containing ¬l are truncated (Line 13).

case D at Lines 15–16, where l has been refuted as a desire. Literal l is removed
from chain for desires and goals (Line 16).

otherwise at Lines 17–18, where \( l \) has been refuted either as a goal, an intention, or a social intention. According to Definition 8, literal \( l \) is removed from chains of the right mode (either G, I, or SI) at Line 18.

5.3. Computational Results

We shall present the computational properties of the algorithms previously described. Since Algorithm 2 **Proved** and Algorithm 3 **Refuted** are sub-routines of the main one, we shall exhibit the correctness and completeness results of these algorithms inside theorems for Algorithm 1 **DefeasibleExtension**.

In order to properly demonstrate results on the computational cost of the algorithms, we need the following definition.

**Definition 23.** Given a finite, modal defeasible theory \( D \), the size \( \Sigma \) of \( D \) is the number of occurrences of literals plus the number of the rules in \( D \).

For instance, the size of the theory

\[
F = \{ a, \mathcal{O} b \},
R = \{ r_1 : a \Rightarrow_{\mathcal{O}} c, \ r_2 : a, \mathcal{O} b \Rightarrow d \}, \ \triangleright = \emptyset
\]

is counted 9 (literal \( a \) counts three times).

We also report some key ideas and intuitions behind our implementation.

1. Each operation on global sets \( \pm \circ \mathcal{O} \) and \( \circ \pm \) requires linear time, as we manipulate finite sets of literals;
2. For each literal \( \Box l \in HB \), we implement an hash table with pointers to rules where the literal occurs; thus, retrieving the set of rules containing a given literal requires constant time.
3. The superiority relation can also be implemented by means of hash tables; once again, the information required to modify a given tuple can be accessed in constant time.

**Lemma 24.** Given a finite, modal defeasible theory \( D \) with size \( \Sigma \), Algorithms 2 **Proved** and 3 **Refuted** terminate and their computational complexity is \( O(\Sigma) \).

**Proof.** Every time Algorithms 2 **Proved** or 3 **Refuted** are invoked, they both modify a subset of the set of rules \( R \), which is finite. Consequently, we have their termination. Moreover, since \( |R| \in O(\Sigma) \) and each rule can be accessed in constant time, we obtain that their computational complexity is \( O(\Sigma) \). \( \square \)
Theorem 25. Given a finite, modal defeasible theory $D$ with size $\Sigma$, Algorithm 1 `DefeasibleExtension` terminates and its computational complexity is $O(\Sigma)$.

Proof. The most important part to analyse concerning termination of Algorithm 1 `DefeasibleExtension` is the repeat/until routine at Lines 14–39. Once an instance of the cycle has been performed, we are in one of the following, mutually exclusive situations:

1. No modification of the extension has occurred. In this case, Line 39 ensures the termination of the algorithm;
2. The theory has been modified with respect to a literal in $HB$. Notice that the algorithm takes care of removing the literal from $HB$ once the suitable operations have been performed (specifically, at Line 3 of Algorithm 2 `Proved` and 3 `Refuted` as well). Since this set is finite, the process described above eventually empties $HB$ and, at the next iteration of the cycle, the extension of the theory cannot be modified. In this case, the algorithm ends its execution as well.

Moreover, Lemma 24 proved the termination of its internal sub-routines.

In order to analyse computational complexity of the algorithm, it is of the utmost importance to correctly comprehend Definition 23. Remember that the size of a theory is the number of all occurrences of each literal in every rule plus the number of the rules. The first term is, usually, (much) bigger than the latter.

Let us examine a theory with $Y$ literals and whose size is $Z$, and consider the scenario when an algorithm $A$, looping over all $Y$ literals of the theory, invokes an inner procedure $P$ which selectively deletes a literal given as input from all the rules of the theory (no matter to what end). A rough computational complexity would be $O(Z^2)$, given that, when one of the $Y \in O(Z)$ literal is selected, $P$ removes all its occurrences from every rule, again $O(Z)$.

However, a more discerning analysis shows that the complexity of $A$ is lower. The mistake being due to the fact that the complexity of $P$ was considered separately from the complexity of the external loop, while instead they are strictly dependent. Indeed, the overall number of operations made by the sum of all loop iterations cannot outrun the number of occurrences of the literals, $O(Z)$, because the operations in the inner procedure directly decrease, iteration after iteration, the number of the remaining repetitions of the outmost loop, and the other way around. Therefore, the overall complexity is not bound to $O(Z) \cdot O(Z) = O(Z^2)$, but to $O(Z) + O(Z) = O(Z)$. 

Ph.D. Thesis - 2014 Francesco Olivieri
5.3. COMPUTATIONAL RESULTS

We can now contextualise the above reasoning to Algorithm 1 Defeasible-Extension, where $D$ is the theory with size $\Sigma$. The initialisation steps (Lines 1–7 and 12–13) add an $O(\Sigma)$ factor to the overall complexity. The main cycle at Lines 14–39 is iterated over $HB$, whose cardinality is in $O(\Sigma)$. The analysis of the precedent paragraph implies that invoking Algorithm 2 Proved at Lines 9 and 31 as well as invoking Algorithm 3 Refuted at Lines 10, 17, 28 and 29 represent an additive factor $O(\Sigma)$ to the complexity of repeat/until routine and for cycle at Lines 8–11 as well. Finally, all operations on the set of rules and the superiority relation require constant time, given the implementation of data structures proposed. That being the case, we can conclude that the complexity of the algorithm is $O(\Sigma)$.

Theorem 26. Algorithm 1 DefeasibleExtension is sound and complete.

Proof. As already argued at the beginning of the section, the aim of Algorithm 1 DefeasibleExtension is to compute the defeasible extension of a given modal theory $D$ through successive transformations on the set of facts and rules, and on the superiority relation: at each step, they compute a simpler theory while retaining the same extension. Again, we remark that the word “simpler” is used to denote a theory with fewer elements in it. Since we have already proved the termination of the algorithm, it eventually comes to a fixed-point theory where no more operations can be made.

In order to demonstrate the soundness of Algorithm 1 DefeasibleExtension, we show in the list below that all the operations performed by the algorithm are justified by Proposition 16 and described in Lemmas 34–48, where we prove the soundness of each operation involved.

1. Algorithm 1 DefeasibleExtension:
   - Lines 2–3 and 7: Lemma 34;
   - Line 9: item 2. below;
   - Line 10: item 3. below;
   - Line 17: Lemma 47 and item 3. below;
   - Line 26: Lemma 46 and item 2. below;
   - Lines 28–29: Lemma 48 and item 3. below;
   - Line 31: Lemma 45 and item 2. below;

2. Algorithm 2 Proved:
   - Line 4: Lemma 48 and item 3. below;
   - Line 5: Part 2. of Proposition 16 and item 3. below;
• Line 6: Part 3. of Proposition 16 and item 3. below;
• Lines 7–9: Lemma 35;
• Case B at lines 11–14: Lemma 37;
• Case O at lines 15–17: Lemma 39;
• Case D at lines 18–22: Lemma 41;
• Otherwise at lines 23–24: Lemma 43;

3. Algorithm 3 Refuted:

• Lines 4–6: Lemma 36;
• Case B at lines 8–11: Lemma 38;
• Case O at lines 12–14: Lemma 40;
• Case D at lines 15–16: Lemma 42;
• Otherwise at lines 17–18: Lemma 44;

The result of these lemmas is that whether a literal is defeasibly proved or not in the initial theory, so it will be in the final theory. This proves the soundness of the algorithm.

Moreover, (i) All lemmas show the equivalence of the two theories, and (ii) The equivalence relation is a bijection; this demonstrates the completeness of Algorithm 1 \textsc{DefeasibleExtension}. \textit{Quod erat demostrandum.} □

Summary

This chapter presented all the algorithms necessary to compute: (i) How the operational environment actually is/works, in the form of “belief literals”; (ii) Which norms are in force in such an environment, in the form of “obligation literals”; (iii) What are the agent’s outcomes, in the form of “desire, goal, intention and social intention literals”. We provided a detailed description on how they behave.

The chapter was ended by the computational results. We proved, in order of presentation, that the algorithms terminate and have a linear computational complexity in the order of the input theory, and then we reported the theorem regarding soundness and completeness. Proof of such a theorem lays on demonstrations of Lemmas 35–48 and Proposition 34 which will be detailed in Appendix B.

The starting point of this chapter was a declarative description given in the logical framework we proposed. Being the algorithms able to compute the positive, and negative extensions of a modal defeasible theory, at the end of the computation we essentially are in one of two mutually exclusive situations:
“Yes”, the theory is compliant with respect both norms and outcomes, or “No” and either some norms are not and cannot be repaired, or some fundamental outcomes cannot be achieved.

Strategies to change the theory in case of a “No” answer are out of the scope of this dissertation and are in the field of business process revision under compliance.

The rest of this dissertation deals with the “Yes” answer, that is how to synthesise a process graph starting from the literals in the modal, positive extension given as output by the algorithms just proposed.
We distinguished three phases an agent must pass through to bring about certain states of affairs: First, understanding the environment she acts in, then the agent deploys such information to deliberate which objectives to pursue and, finally, how to act in order to reach them.

In the first phase, the agent gives a formal declarative description of the environment (in our case, a rule-based formalism). In the second stage, the agent combines the formal description with an input describing a particular state of affairs of the environment, and determines which norms are actually in force, which objectives she decides to commit to and to which degree. The agent’s decision is based on logical derivations. Since the agent’s knowledge is represented by rules, during the last phase, the agent exploits information from a derivation to select the activities to carry out in order to achieve the objectives. We pointed out that a derivation can be understood as a virtual simulation of the various activities involved.

To face the first two problems, in the previous chapters we proposed a modal logic, and gave algorithms to compute which literals can be derived in a specific context determining the environment the agent is situated in.

We now study how to determine the courses of action the agent may undertake in order to fulfil her objectives after both compliances have been established. To this end, we take inspiration from Business Process Modelling.

We described a business process as a collection of related, structured tasks that produce a specific service, where a task is an activity that needs to be accomplished within a defined period of time [Davenport, 1993]. A business
process can be understood as the set of all its possible execution traces, i.e., all the possible ways in which the process can be executed, in terms of linear sequences of tasks/actions. Notice that a trace is equivalent to a classical AI plan [Ghallab et al., 2004].

A derivation of a given (outcome) literal corresponds to a specific course of action leading to that particular outcome, the reason being that the information is encoded in the agent’s knowledge base. Thus, we can compare such derivations to traces.

In a business process, there may be multiple, possible ways to reach a given outcome while remaining norm compliant. If this is the case, it is pointless for the agent to perform all the alternatives to bring about the same state of affairs. But it is nonetheless important that these pieces of information are not lost: The agent should be equipped with a process graph showing all such alternatives and allowing the agent to carry out the best strategy. Notice that choosing the best strategy relies on external information that does not affect the construction of the process graph, such as risks, task cost, execution time, minimum number of task to perform, and so on. Studying best strategy analysis is out of the scope of the present doctoral dissertation.

The problem we shall address in this chapter is how to transform the logic describing the organisation and the environment into a process graph. This process graph is structured with the typical BPMN gateways (parallel and choice) and respects all the execution traces.

This chapter is based on [Olivieri et al., 2013].

Outline

The present chapter discusses the following contents. We shall start by explaining the underlying ideas behind the algorithms. We shall delve into the problematics spotted during our studies, and briefly explain how they were tackled (Section 6.1). Then, we shall proceed by first introducing the notation adopted by the algorithms (Section 6.1.1), and then by proposing the algorithms themselves. Each algorithm is followed by a detailed account on how they operate (Section ??). We shall end by presenting the computational properties (Section 6.4); specifically, the computational cost of the proposed solutions and a proof of their correctness, whereas the algorithms are not required to be complete.
6.1. Principles underlying our approach

We illustrate how to model a process graph starting from a modal defeasible theory describing a compliant situation. Accordingly, the process graph resulting by the execution of the main algorithm, namely Algorithm 4 \textsc{complianceByDesign}, will be compliant by design. Here, we choose intention as the mental attitude the agent has to comply with, and consequently the theory describing the agent capabilities derives \( -\partial_O \bot \) as well as \( -\partial_I \bot \). We do not opt for social intentions for illustrative purposes; thus we give the agent a wider choice of possibilities whether to comply with respect to a specific norm.\(^1\) The algorithm can be easily modified to treat outcome compliance with respect to the other mental attitudes\(^2\).

Follows the intuitive idea of our approach.

**First phase:** Select the first element proved as intention in the chain of each intention-compliant rule (such literal being \( u_i \) in the algorithms). Elements in such chains are ranked according to the agent’s preference order and as such represent acceptable alternatives for the agent. For each of such literals, we create a new node (if not already present in the graph), and a X-Or Split/Join pattern among \( u_i \) and all the elements in the chain following it which are desires and/or intentions with a factual derivation (that is elements in either \( +\partial_D \) or \( \partial_I \), as well as in \( +\partial_B \)). In this way, we are transforming literals in such chains as the tasks in between the X-Or Split and the X-Or Join. Being those elements the final objectives the organisation wants to achieve, the X-Or Join is attached to the End node.

The X-Or Split has an incoming edge for each belief rule proving one of the literals involved in such X-Or pattern. We propose the exclusive choice pattern among elements since achieving any one of these elements implies the outcome compliance with respect to that particular rule/chain.

**Second phase:** Navigate backwards the derivation tree, rule by rule, until the facts of the theory are met. We propose a backward approach because Algorithm 1 \textsc{defeasibleExtension} has already calculated the positive and negative extensions of the input theory. Thus, we can now distinguish, and consequently consider, only those rules with both conclusion and set of antecedents in the positive extension.

\(^1\) Indeed, many scenarios see the “best” option for the agent not to comply with a specific norm. Let us assume that by doing task \( A \) the agent incurs a relatively small sanction, which does not foresee any major crime. What should the agent do if the sanction is of \( X \) euros but the benefits of doing \( A \) are of \( Y \) euros, with \( Y \) significantly bigger than \( X \)?

\(^2\) Specifically, by simply modifying the assignment \( u_i \in +\partial_I \) at Line 5 of Algorithm 4.
Consequently, for a given literal $l$, we create a new node and collect all the rules proving it. Such rules are the last step of all possible derivations (and so traces) through which the agent can bring about the state of affairs described by $l$. By performing all the steps of one of these alternatives, the agent “gets” $l$. These multiple ways can be naturally linked together through an Or Join node, given that a node with an incoming edge from the Or Join node is allowed to start its execution while one of the nodes with an outgoing edge to the Or Join node has been successfully executed.

Third phase: We recall that a rule consists of a conclusion and a set of premises: each premise must be individually proved for the rule to be applicable (as stated in Definitions 3, 5, and 7). Again, this property has a natural counterpart in the And Join node among the premises, where all nodes with an outgoing edge to the And Join node must separately complete their execution before a node attached with an incoming edge from the And Join node is allowed to start its run. This strategy is iterated backwards for each antecedent of each rule for $l$ and is processed by Algorithm 5 backTrack.

Fourth phase: Once this backwards procedure ends, the process graph can be synthesised. This is done in three subsequent steps. We point out that the solution we propose is not required to be optimal: it is out of the scope of this dissertation to investigate when a synthesis may be called “better” than another one. Nonetheless, we think that the proposed procedures represent reasonable techniques to simplify a graph by grouping nodes from the point of view of parallel and choice patterns, as well as conditions elimination.

Fourth phase—First step: We identify the co-occurrence of literals. We say that a set of literals co-occurs if, and only if, whenever one of its elements is in the antecedent of a given rule, then all the other elements are in the antecedent as well. Formally,

**Definition 27** (Literals co-occurrence). Let $D = (F, R, >)$ be a defeasible theory. We say that two distinct literals $a$ and $b$ co-occur if $\exists R' \subseteq R, |R'| \geq 2$, such that $\forall r \in R'. a \in A(r)$ iff $b \in A(r)$.

Notice that it is easy to verify that the co-occurrence is an equivalence relation over literals. Thus, given a set $S$ of co-occurring literals, the idea is to represent literals in $S$ as a separate building block able to interact with the other parts of the graph. Anyhow, this operation makes sense from a computational point of view only when a co-occurrence set is present strictly in more than one rule.

Fourth phase—Second step: We recognise Split patterns (Algorithm 6 split-PATTERN). This step is needed because literals may occur together in more than
one antecedent without co-occurring. One can be tempted to solve the problem by a simple incremental approach, that is by identifying those literals which appear together with more frequency, grouping them, and iterating the step on the remaining nodes.

Let us consider the theory with the following rules, where we adopt the notation (used in the rest of the dissertation) that a literal \( u_i \) stands for an outcome to achieve.

**Example 11.**

\[
\begin{align*}
  r_1 : a, b &\Rightarrow u_1, \\
  r_2 : a, b, c &\Rightarrow u_2, \\
  r_3 : a, b, c, d &\Rightarrow u_3.
\end{align*}
\]

The result of this *modus operandi* is depicted in Figure 6.1.

![Figure 6.1: Instance of Split pattern recognition: Correct synthesis.](image)

Here, we group A and B to produce \( U_1 \), but the combination of A and B contribute to bring about \( U_2 \) as well as \( U_3 \). Accordingly, we add an Or Split node (top-right node, blue background), which enters \( U_1 \). The Or Split node is subsequently linked to C through an And Join, which in turn is linked to another Or Split (bottom-right node, blue background) to get \( U_2 \). In turn, this Or Split and D enter an And Join to get \( U_3 \).

However, this method fails under certain configurations of literals, as pointed out by the following example.

**Example 12.**

\[
\begin{align*}
  r_4 : e, f, g &\Rightarrow u_4, \\
  r_5 : e, f, h &\Rightarrow u_5, \\
  r_6 : f, g, i &\Rightarrow u_6.
\end{align*}
\]

The solution obtained by applying the incremental approach is incorrect since \( U_4 \) would be obtained twice. In fact, when the execution of the process begins, a token leaves the Start node and enters E, F, and G. Now, the first block of
two nodes is executed in parallel (thus we have two possible traces: EF and FE). Same happens for the second block where the two possible traces (green background) are FG and GF. Finally, the AND JOIN in the green background produces either EFG or FEG, while at the same time the bottom AND JOIN in the green background produces either FGE or GFE. This is incorrect given that in the corresponding theory of Example 12 only one derivation (execution) of \( e \), \( f \), and \( g \) is needed to obtain \( U_4 \).

We illustrate the solution adopted in Algorithm 6 splitPattern. Given a set of antecedents where no set of literals co-occurs, we choose a pivot literal (resp., \( sL \) and \( l \) in the algorithms), and then we identify a set \( S \) such that: i) \( S \) is contained in at least two sets of \( sL \), ii) For each pair of subsets in \( sL \), their intersection is either \( l \) or \( S \).

By satisfying condition ii), we are selecting a subset of literals that “locally co-occurs”, i.e., there are no two distinct, proper subsets of \( S \) which are in two different subsets of \( sL \) (excluding \( S \) itself). Finally, we need to ensure “some constraints of minimality” on \( S \). This is so because we want to “build” \( S \) incrementally (as clarified by Example 14), even if setting \( S \) to be minimal would be incorrect. In fact, given the two sets \( X = \{a, b, c\} \) and \( Y = \{a, b, d\} \), here \( S \) would be minimal when equals either to \( \{a\} \), or \( \{b\} \), while we want \( S \) to be \( \{a, b\} \).

**Fourth phase–Third step:** Based on the idea that a trace is a sequence of only tasks while conditions should be used to annotate tasks or connections among

---

**Figure 6.2.:** Two instances of Split pattern recognition: Wrong synthesis.
tasks, we remove nodes representing conditions as well as nodes representing modal literals; they will become labels of annotated edges. This is motivated by the fact that we are considering only literals in the positive extension of the theory. That being the case, every condition necessary for the task to start its execution is satisfied. Even when we consider modal literals such as obligations, these have been fulfilled as well. A different approach would be not to annotate the edges, but instead to pile up all such literals in a stack, invoke the algorithm for each one of them (considering them as new “sub-outcomes” to reach) and create the corresponding sub-process graph, and finally link it to the main process graph.

Performing this phase of the algorithm guarantees that no dependency between a modal literal in the antecedent of a rule and the corresponding conclusion of such a rule is lost.

### 6.1.1. Algorithm notation

As already pointed out, we model the collection of possible alternative courses of action as a process graph $G = (V, E)$, named ComplianceGraph in the algorithms, where $V$ is a set of nodes and $E$ a set of edges. Nodes represent relevant tasks, while (possibly labelled) edges represent logical dependency among literals, based on rules of the initial theory.

Each literal $l \in \text{Lit}$ has a unique counterpart in the graph, namely a node labelled with $L$. In notation, the literal is a lowercase letter, while the node is denoted by a small capital letter. This will ease the definition of the formal properties and operations in the algorithms; then we shall be able to talk about a given node and then refer to the corresponding literal in the theory (and the other way around) without formally bind them every time.

At the end of the execution of the algorithms all conditions such as $I_a_1$ or $O_a_2$ will be represented as labels attached to edges. Regardless of this, during the execution of the algorithms such conditions will be temporarily represented by nodes. For instance, $I_a$ will be depicted by the node INT A$_1$, while $O_b$ will be OBL A$_2$.

For any node $X \in V$, we define $\text{inv}_V(X) = \{ Y \in V \mid (Y, X) \in E \}$ and $\text{out}_V(X) = \{ Y \in V \mid (X, Y) \in E \}$. Such structures will ease the retrieval of all nodes linked to a specific one.

We shall use a semi-formalised language, a pseudocode, that makes use of reserved words. In particular we use the word *procedure* to refer to any kind of subprogram in the code. Moreover, we use the two reserved words *Exit* and
Break with the following semantics. The Exit command ends the actual run of the algorithm, while the Break command ends the computation of the loop construct where the command appears.

Algorithms work on active rules. We define a rule to be active rule if (i) Is applicable, (ii) Contributes to derive a literal in the positive extension, and (iii) Is not defeated by any applicable rule for the opposite.³ Formally

Definition 28 (Active rules). Let \( D = (F, R, >) \) be a modal, defeasible theory and \( E^+(D) \) the corresponding positive, extension. We say that a given rule \( r \in R^B \) is active if:

(i) \( r \) is applicable,
(ii) \( C(r) \in E^+(D) \), and
(iii) \( \not\exists s \in R \) such that \( s \) is applicable and \( s > r \).

We do not distinguish between Or Split and And Split: We consider them in the graph simply as Or Split (Or-S in notation). On the contrary, And Join and Or Join are distinct from one another.

6.2. Algorithmic results for the process graph synthesis

Follows algorithms for the process graph synthesis along with their description. We begin with Algorithm 4 complianceByDesign, the startup of the entire computation.

Algorithm 4 complianceByDesign starts by selecting the set of active rules (Lines 1–3).

The for cycle at Lines 5–23 initialises the backtrack procedure briefly described above. Line 6 selects the compliant choices for each active outcome rule. If the step picks more than one element (if condition at Line 7), then we represent them with an X-Or split/join pattern (then branch at Lines 8–14), otherwise no additional construction is needed (Lines 19–20).

Depending on the number of outcomes selected by Line 7, Algorithm 5 backTrack is invoked with a different node as second parameter (Lines 16 and 21). In both cases, the computation only considers the factual part of the rules leading to the current outcome \( u_j \).

Figure 6.3 shows the graph resulting from the execution of Algorithm 5 backTrack.

³We choose to select only non-defeated rules for beliefs. Another option would be to consider all rules proving a given literal in the positive extension, given that the whole set forms a team defeater against rules for the opposite conclusion. Our motivation to exclude the “weak members” from the set was to limit the actual choices in order to give to the agent the set of the strongest (best) candidates.
Algorithm 4 complianceByDesign (Part 1)

Input: A modal defeasible theory $D$ and its extension $E(D)$
Output: A process graph $G$

1: $R_{ACT} \leftarrow \{ r \in R^\Box \cup R^B : C(r) \subseteq +\partial \}$
2: $R_{ACT} \leftarrow R_{ACT} \setminus \{ r \in R_{ACT} : \exists a \in A(r) \text{ such that } a \in -\partial \}$
   \hspace{1cm} \triangleright \text{ if } r \in R^B \text{ then } \square = \Box \text{ and } \exists a \in A(r) \text{ such that } a \in -\partial \$
3: $R_{ACT} \leftarrow R_{ACT} \setminus \{ r \in R_{ACT} : s > r, s \in R_{ACT} \}$
4: ComplianceGraph $= (V = \{ \text{start, end}\}, E = \emptyset)$
5: for $r \in R^U \cap R_{ACT}$ such that $\exists u_i \in C(r)$ such that $u_i \in +\partial_1$ and $\forall u_j, j < i, u_j \notin +\partial_1$ do
   6: $goals \leftarrow \{ u_j \in C(r) | j \geq i, u_j \in +\partial_B \text{ and } (u_j \in +\partial_1 \text{ or } u_j \in +\partial_D) \}$
   7: if $|goals| \geq 2$ then
      8: $V \leftarrow V \cup \{ \text{XOR-S}_r \} \cup \{ \text{XOR-J}_r \}$
      9: $E \leftarrow E \cup \{ (\text{XOR-J}_r, \text{END}) \}$
   10: for $u_j \in goals$ do
          11: $V \leftarrow V \cup \{ U_j \}$
          12: $E \leftarrow E \cup \{ (\text{XOR-S}_r, U_j) \}$
          13: $E \leftarrow E \cup \{ (U_j, \text{XOR-J}_r) \}$
    14: end for
    15: for $u_j \in goals$ do
    16: backTrack($u_j, \text{XOR-S}_r, \{ r \in R_{ACT} \cap R^B | r \text{ proves } u_j \}$)
    17: end for
    18: else
      19: $V \leftarrow V \cup \{ U_j \}$
      20: $E \leftarrow E \cup \{ (U_j, \text{END}) \}$
      21: backTrack($u_j, U_i, \{ r \in R_{ACT} \cap R^B | r \text{ proves } u_i \}$)
      22: end if
23: end for
24: $V' \leftarrow V$
25: for $L \in V'$ do \hspace{1cm} \triangleright \text{ Co-occurrence}
26: $cO \leftarrow \{ M \in V' | out_V(L) = out_V(M) \}$
27: if $|out_V(L)| \geq 2 \text{ and } |cO| \geq 2$ then
      28: $V \leftarrow V \cup \{ \text{AND-J}_L \} \cup \{ \text{OR-S}_L \}$
      29: $E \leftarrow E \cup \{ (M, \text{AND-J}_L) | M \in cO \} \cup \{ (\text{AND-J}_L, \text{OR-S}_L) \}$
      \hspace{1cm} $\cup \{ (\text{OR-S}_L, N) | N \in out_V(L) \} \setminus \{ (M, N) | M \in cO \text{ and } N \in out_V(L) \}$
   30: end if
31: $V' \leftarrow V' \setminus cO$ \hspace{1cm} \triangleright \text{(Continue)}
Algorithm 4 complianceByDesign (Part 2)

33: Let $V$ be the list of the literals with a corresponding node in $V$
34: for $l \in V$ such that $|\text{out}(L)| \geq 2$ do $\triangleright$ Split pattern
35: $sL \leftarrow \{A(r) | r \in R_{ACT} \text{ and } l \in A(r)\}$
36: $\text{candGlobal} \leftarrow \emptyset$
37: $\text{splitPattern}(l, sL, \text{null})$
38: $V \leftarrow V \setminus \text{candGlobal}$
39: Remove $l$ from $V$
40: end for
41: while $\exists L \in V$ such that: i. $L$ is an And-J node and $|\text{in}_V(L)| = 1$, ii. $l$ is a condition, or iii. $l = \Box m$, with $\Box \in \text{MOD} \setminus \{B\}$ do $\triangleright$ Contraction
42: $E \leftarrow E \cup \{e = (X, Y) | (X, L) \in E \text{ and } (L, Y) \in E \} \setminus \{(X, Y)| \text{either } X = L, \text{ or } Y = L\}$
43: $\text{label}(e) \leftarrow \{l\} \cup \text{label}((X, Y))$ with $X = L$ or $Y = L$
44: $V \leftarrow V \setminus \{L\}$
45: end while
46: return $G$

Track with input theory $D$ of Example 13. In the figures we consider only the subprocesses connecting the facts to the outcomes ignoring Start and End nodes.

The for cycle at Lines 24–32 recognises the co-occurrence of nodes. This modification is necessary only when they appear in more than one antecedent (if condition at Line 27). Here, we consider only literals with more than one outgoing edge ($|\text{out}_V(L)| > 2$). For each set of co-occurrent literals, we create an And-J node and Or-S node (Line 28), and we adjust the graph accordingly (Line 29).

The algorithm now synthesises the Split patterns (Lines 34–40). Line 33 selects only the nodes representing literals; $sL$ is the set of antecedents where the current literal $l$ appears in (Line 35).

Finally, we remove nodes representing conditions and modal literals by substituting the corresponding node and edges attached to it with an unique labelled edge. This process combined with the synthesis of literal co-occurrence and Split pattern may have created And-J nodes with only one incoming edge: they need to be erased. The while cycle at Lines 41–45 implements these two operations.

Algorithm 5 backTrack performs the backwards construction of the process graph. It receives three inputs: (i) The last processed literal ($l$), (ii) The set of active rules to be considered ($R_{bT}$), (iii) The node (N) that will be attached to the edges introduced by the current run of the algorithm. Such a node is either itself, or the XOR-S whenever $l$ is in the chain of an outcome rule.
Algorithm \(5\) \texttt{backTrack}  

1: \textbf{procedure} \texttt{backTrack}(\textit{l}, \text{n}, set of rules \(R^{bT}\))  
2: \hspace{1em} \textbf{if} \( l \in V \) \textbf{then} \texttt{Exit}  
3: \hspace{2em} \( R_{ACT} \leftarrow R_{ACT} \setminus R^{bT} \)  
4: \hspace{1em} \textbf{if} \( l \in F \) \textbf{then}  
5: \hspace{2em} \( E \leftarrow E \cup \{(\text{start}, N)\} \)  
6: \hspace{2em} \texttt{Exit}  
7: \hspace{1em} \textbf{end if}  
8: \hspace{1em} \textbf{if} \( R^{bT} = \{r\} \) and \( A(r) = \{a\} \) \textbf{then}  
9: \hspace{2em} \( V \leftarrow V \cup \{A\} \)  
10: \hspace{2em} \( E \leftarrow E \cup \{(A, N)\} \)  
11: \hspace{2em} \texttt{backTrack}(a, A, \{r \in R_{ACT} \cap (R^D \cup R^B) | a \in C(r)\})  
12: \hspace{2em} \textbf{end if}  
13: \hspace{1em} \texttt{Exit}  
14: \hspace{1em} \textbf{end if}  
15: \hspace{1em} \textbf{if} \( |R^{bT}| > 1 \) \textbf{then}  
16: \hspace{2em} \( V \leftarrow V \cup \{\text{Or-J}_N\} \)  
17: \hspace{2em} \( E \leftarrow E \cup \{(\text{Or-J}_N, N)\} \)  
18: \hspace{2em} \texttt{for} \( r \in R^{bT} \) \texttt{do}  
19: \hspace{3em} \textbf{if} \( A(r) = \{a\} \) \textbf{then}  
20: \hspace{4em} \( V \leftarrow V \cup \{A\} \)  
21: \hspace{4em} \( E \leftarrow E \cup \{(A, \text{Or-J}_N)\} \)  
22: \hspace{4em} \texttt{backTrack}(a, A, \{r \in R_{ACT} \cap (R^D \cup R^B) | r \text{ proves } a\})  
23: \hspace{3em} \textbf{else}  
24: \hspace{4em} \( V \leftarrow V \cup \{\text{And-J}_r\} \)  
25: \hspace{4em} \( E \leftarrow E \cup \{(\text{And-J}_r, \text{Or-J}_N)\} \)  
26: \hspace{4em} \texttt{for} \( a \in A(r) \) \texttt{do}  
27: \hspace{5em} \( V \leftarrow V \cup \{A\} \)  
28: \hspace{5em} \( E \leftarrow E \cup \{(A, \text{And-J}_r)\} \)  
29: \hspace{5em} \texttt{backTrack}(a, A, \{r \in R_{ACT} \cap (R^D \cup R^B) | r \text{ proves } a\})  
30: \hspace{4em} \texttt{end for}  
31: \hspace{3em} \texttt{end if}  
32: \hspace{1em} \texttt{end for}  
33: \hspace{1em} \texttt{Exit}  
34: \hspace{1em} \textbf{end if}  
35: \hspace{1em} \textbf{end procedure}
If literal \( l \) has already been analysed, and therefore node \( L \) has already been added to the graph, then no operations are needed (Line 2).

The algorithm performs different operations depending on: (i) How many rules prove \( l \), and (ii) The number of elements in the antecedent. If \( l \) is a fact (Lines 4–7), then we link \( N \) to \( \text{START} \). Otherwise, we create a node for each element in the antecedent not already in \( V \). If the antecedent contains more than one element, we add an \( \text{And–J} \) node between the nodes representing the antecedent and \( N \) (if condition at Line 14; \textbf{else} at Line 32).

Furthermore, if \( l \) is derived by more than one rule, the algorithm adds an \( \text{Or–S} \) node and links it to the suitable nodes (if condition at Line 24). The algorithm invokes itself on every antecedent of every rule found (Lines 11, 20, 31, and 38).

To explain the behaviour of the algorithms we propose two examples. The first one is reported below, while the second one is postponed after Algorithm 6 \textsc{splitPattern}. In fact, theory \( D_1 \) in Example 13 is useful to illustrate

1. the initial steps of Algorithm 4 \textsc{complianceByDesign} when it initialises the X-Or structures of feasible outcomes,
2. how Algorithm 5 \textsc{backTrack} works,
3. co-occurrence and labelling of edges,

while it is deliberatively not intended to show \textsc{split} pattern recognition, which is left to Example 14. For simplicity, we assumed all rules of \( D_1 \) to be active. Figure 6.3 illustrates points 1. and 2. the, Figure 6.4 illustrates point 3.

**Example 13.** Let \( D_1 = (\Delta, \Gamma \{t_1, \ldots, t_5, c_1\}, R, \emptyset) \) be a modal, defeasible theory such that

\[
\begin{align*}
R &= \{ r_1 : \Delta \Rightarrow_U u_1 \odot u_2 \odot u_3 \} & r_2 : \Gamma \Rightarrow_U \neg u_2 \\
& r_3 : t_1, t_2, t_3 \Rightarrow u_1 & r_4 : t_4, c_1 \Rightarrow u_1 \\
& r_5 : t_1, t_2, \text{O}o_1 \Rightarrow u_3 & r_6 : t_4, t_5 \Rightarrow_{\text{O}} o_1 \}.
\end{align*}
\]

The positive extension \( E^+(D_1) \) is as follows (the notation is that of Algorithm 1):

\[
E^+(D_1) = \{ + \partial_B = \{ \Gamma, \Delta, t_1, \ldots, t_5, c_1, u_1, u_3 \} \\
+ \partial_\text{O} = \{ o_1 \} \\
+ \partial_\text{D} = \{ u_1, u_2, \neg u_2, u_3 \} \\
+ \partial_\text{G} = \{ u_1 \} \\
+ \partial_\text{I} = \{ u_1 \} \\
+ \partial_\text{SI} = \{ u_1 \}\}.
\]
Labels of the edges for $c_1$ and $O_o_1$ are highlighted by dotted rectangles; the co-occurrence of tasks $T_1$ and $T_2$ is highlighted by a dashed rectangle (notice that preconditions $\Gamma$ and $\Delta$ do not appear in the figure since they are not significant for the process graph).

**Figure 6.3.** Process graph of theory $D_1$ resulting from Algorithm 4 **COMPLIANCE-ByDesign**: Run of the algorithm up to Line 23

First, the X-Or pattern is made for node $U_1$ and $U_3$. Notice that: (i) $O_o_1$ is represented by OBL $O_1$, (ii) $U_2$ is not included since $u_2 \notin +\partial_B$, and (iii) $r_2$ is intention-compliant since $C(r_2) \notin +\partial_l$. Since there are two rules proving $u_1$, Or-J node is added with the unique label indicating which rules are involved (namely, $r_3$ and $r_4$).

The next step is to evaluate $A(r_3)$. This set contains more than one element and thus AND-J$r_3$ is added to $V$. To that node, we attach nodes $T_1$, $T_2$ and $T_3$. When Algorithm 5 **backTrack** is invoked on either $t_1$, $t_2$ or $t_3$ no further operations are needed since they are all facts of $D_1$. The steps necessary to backtrack for $r_4$ are the same as those just described. The algorithm returns from the call on the rules proving $u_1$, and now passes to evaluate the only rule proving $u_3$, that being $r_5$. Antecedents of $r_5$ are $t_1$, $t_2$, and $O_o_1$. No operations are made for the first two elements given that they have already been processed. $O_o_1$ is proved by $r_6$ which has two antecedents: $t_4$ and $t_5$. Accordingly, AND-J$r_6$ is added to $V$.

The only co-occurent set of literals is $\{t_1, t_2\}$. This set is present in more than one antecedent, satisfying if test at Line 27. Accordingly, nodes AND-J$T_1,T_2$ and OR-S$T_1,T_2$ are added to $V$ (and linked together), while all outgoing edges from
T₁ and T₂ are erased from E and replaced by (T₁, AND-J₁, T₂), (T₂, AND-J₁, T₂),
(AND-J₁, OR-S₁, T₂), (OR-S₁, AND-J₁), and (OR-S₁, AND-J₁).

Finally, the main algorithms erase conditions and modal literals (c₁ and O₁) by labelling edges. Thus, edges (START, C₁) and (C₁, AND-J₁) are replaced by (START, AND-J₁) which is labelled with C₁. Same thing happens for edges (AND-J₁, OBL O₁) and (OBL O₁, AND-J₁) which are substituted by the single edge (AND-J₁, AND-J₁) with label OBL O₁.

Algorithm 6 SPLITPATTERN takes three parameters as input. The third one distinguishes whether the algorithm was invoked either (i) by Algorithm 4 COMPLIANCEBYDESIGN at Line 37, or (ii) recursively at Line 42; if (ii), the OR-node input will be attached to the AND-J node created at Line 22.

First, the algorithm computes set S: this is done from Line 3 to Line 18. At Line 3 we initialise a support set apt to store all the literals (without repetition) in each \( sL \). For each of such elements, we select the sets of antecedents where it appears in, and we intersect them. This intersection will be crucial afterwards, specifically at Line 14.

Apart from the pivot literal \( l \), we consider as candidate members to be in S only those literals which occur in at least two antecedents (Line 8). If no such literal exists, then the current run of the algorithm is a recursive invocation and we set S to the singleton \( \{ l \} \) (Line 10). Otherwise, we select the new co-pivot \( a \). At this point a possible choice would be to set S to \( \{ l, a \} \). But it may be the case that \( a \) “locally co-occurs” with other literals except \( l \) (by locally we
6.2. ALGORITHMIC RESULTS FOR THE PROCESS GRAPH SYNTHESIS

Algorithm 6 splitPattern

1: procedure splitPattern(literal \( l \), set \( sL \), node Or-node)  
2: \( \text{candGlobal} \leftarrow \text{candGlobal} \cup \{l\} \)  
3: \( \text{elems}_L \leftarrow (\bigcup_{a \in A(r), A(r) \in sL} a) \)  
4: for \( a \in \text{elems}_L \) do  
5: \( \text{sets}_A_a \leftarrow \{A(r) \in sL | a \in A(r)\} \)  
6: \( \text{intersec}_a \leftarrow \bigcap \text{sets}_A_a \)  
7: end for  
8: \( \text{candidates} \leftarrow \{a \in \text{elems}_L \setminus \{l\} | |\text{sets}_A_a| \geq 2\} \)  
9: if \( \text{candidates} = \emptyset \) then  
10: \( \text{candidates} \leftarrow \{l\} \)  
11: end if  
12: for \( a \in \text{candidates} \) do  
13: \( \text{candGlobal} \leftarrow \text{candGlobal} \cup \{a\} \)  
14: \( S' \leftarrow \bigcap_{b \text{intersec}_a} \text{intersec}_b \)  
15: if \( a \in S' \) then  
16: \( S \leftarrow S' \cup \{l\} \)  
17: else  
18: \( S \leftarrow \{a, l\} \)  
19: end if  
20: \( \text{supp}_L \leftarrow \{A(r) \in sL | S \subseteq A(r)\} \)  
21: \( sL \leftarrow sL \setminus \text{supp}_L \)  
22: \( V \leftarrow V \cup \{\text{And-J}_S\} \cup \{\text{Or-S}_S\} \)  
23: \( E \leftarrow E \cup \{\{\text{And-J}_S, \text{Or-S}_S\}\} \)  
24: if Or-node \( = \) null then  
25: \( E \leftarrow E \cup \{(\text{Or-node}, \text{And-J}_S)\} \)  
26: end if  
27: for \( n \in S \) do  
28: \( E \leftarrow E \setminus \{(N, X) | X = \text{And-J}_r \text{ such that } A(r) \in \text{supp}_L\} \)  
29: \( E \cup \{(N, \text{And-J}_S)\} \)  
30: end for  
31: \( \text{onlyS} \leftarrow \{A(r) \in \text{supp}_L | (\bigcup_{A(t) \in \text{supp}_L, t \neq l} A(t)) \cap A(r) = S\} \)  
32: for \( A(r) \in \text{onlyS} \) do  
33: \( \text{candGlobal} \leftarrow \text{candGlobal} \cup A(r) \)  
34: \( E \leftarrow E \cup \{(\text{Or-S}_S, \text{And-J}_r)\} \)  
35: end for  
36: \( sL_{rest} \leftarrow \text{supp}_L \setminus \text{onlyS} \)  
37: if \( sL_{rest} = \emptyset \) then  
38: \( \text{candidates} \leftarrow \text{candidates} \setminus \text{candGlobal} \)  
39: Break  
40: else  
41: \( sL_{rest} \leftarrow \{A(r) | S \ni A(r) \in sL_{rest}\} \)  
42: choose \( m \in \bigcap_{A(r) \in sL_{rest}} A(r) \)  
43: \( \text{splitPattern}(m, sL_{rest}, \text{Or-S}_S) \)  
44: end if  
45: \( \text{candidates} \leftarrow \text{candidates} \setminus \text{candGlobal} \)  
46: end for  
47: end procedure
CHAPTER 6. CONCERNING PROCESS SYNTHESIS ALGORITHMS

intend restricted to the sets in \(\text{sets}A_d\). For instance, consider the following scenario: \(\{l, a, b, c\}, \{l, a, b, d\}\), where the best choice is not to select \(S = \{l, a\}\) but \(S = \{l, a, b\}\). This is done at Line 14, where we may have the side-effect that \(a\) is not in the resulting intersection. If so, we force \(a\) in \(S\) (Line 18), otherwise we just need to add \(l\) to \(S'\) (Line 16).

After having created (i) the set containing only the antecedents involved in \(S\) (Line 20), and (ii) two unique nodes \(\text{And–J}\) and \(\text{Or–S}\) for \(S\), the algorithm links each element in \(S\) with \(\text{And–J}_S\) (Lines 27–29), which is in turn attached to \(\text{Or–S}_S\) (Line 23).

We now need to discover if there is a subset of the remaining nodes in \(\text{elem}_{\text{L}}\) but not in \(S\) over which it is possible to recursively invoke the \text{splitPattern} procedure. This is done by comparing elements in \(\text{only}_{S}\) with those in \(\text{sL}_{\text{rest}}\) (Lines 30, 35, and 40). If Line 41 selects an \(m\), then Algorithm 6 \text{splitPattern} is recursively invoked (Line 42).

Figures 6.5 and 6.6 show the execution of Algorithm on theory \(D_2\) of Example 14, where Algorithm 6 \text{splitPattern} effectively modifies the graph.

Example 14. Let \(D_2 = (\{t_1, \ldots, t_7, \text{Iu}_1, \ldots, \text{Iu}_5\}, R, \emptyset)\) be a theory such that

\[
\begin{align*}
\mathcal{R} = \{r_1 : t_1, t_2 \Rightarrow \text{u}_1 & \quad r_2 : t_1, t_2, t_3 \Rightarrow \text{u}_2 \\
r_3 : t_1, t_4, t_5 \Rightarrow \text{u}_3 & \quad r_4 : t_1, t_4, t_6 \Rightarrow \text{u}_4 \\
r_5 : t_1, t_5, t_7 \Rightarrow \text{u}_5 & \quad r_6 : t_1, t_2, t_3, t_8 \Rightarrow \text{u}_6 \}.
\end{align*}
\]

We do not report the extension of \(D_2\), nor all the passages of the algorithm to obtain the graph given in Figure 6.6 by starting from the graph of Figure 6.5. These are reported in detail in Appendix B. Again, all rules are active and, to focus only on operations made by Algorithm 6 \text{splitPattern}, no X-Or patterns are present.

6.3. Running examples - Part II

We show how the theories described by the running examples of Section 4.4 are transformed into process graphs. We report the associated theories before

---

\(^4\)What we are trying to catch here is a subtle notion: \(S\) needs to undergo some sort of minimality constraint, while at same time including as many elements as possible. In fact, referring to the previous theory, setting \(S\) to be \(\{a, b\}\) would be an incorrect choice, given that the algorithm would have created only two more nodes. By forcing instead the procedure to look for what we called “local co-occurrences”, the algorithm creates a better synthesis of the \(\text{And}/\text{Or}\) patterns given that the algorithm collects and links more elements together.

Naturally, this method does not pretend to be optimal or to find the optimal solution, but nonetheless we wanted to report the algorithm in this form because we believe that this is the right path for future analysis to improve it.
Figure 6.5.: Process graph of theory $D_2$ resulting from Algorithm 4 COMPILANCE-ByDesign: Run of the algorithm up to Line 32.

Figure 6.6.: Process graph of theory $D_2$ resulting from Algorithm 4 COMPILANCE-ByDesign: Final process graph.
showing the graphs. We start with eye glasses manufacturer example.

\[
F = \{\text{lenses, frames, new\_safety\_regulation}\}
\]

\[
R = \{r_1 \Rightarrow \text{eye\_Glasses}
\]

\[ r'_2 : \text{goggles} \Rightarrow \text{laser} \]

\[ r_3 : \text{lenses, laser} \Rightarrow \text{glasses} \]

\[ r_4 \Rightarrow \text{mounting\_machine1} \]

\[ r_5 \Rightarrow \text{mounting\_machine2} \]

\[ r_6 : \text{mounting\_mach1} \Rightarrow \neg\text{mounting\_mach2} \]

\[ r_7 : \text{glasses, mounting\_machine1} \Rightarrow \text{eye\_Glasses} \]

\[ r_8 : \text{glasses, mounting\_machine2} \Rightarrow \text{eye\_Glasses} \]

\[ r_9 : \text{new\_safety\_regulation} \Rightarrow \neg\text{\_laser} \otimes \text{goggles} \]

\[ r_{10} \Rightarrow \text{\_mounting\_machine1} \otimes \text{\_mounting\_machine2} \]

\[
>^\text{sym} = \{r_6 > r_5\}.
\]

We recall that we use rectangles to describe tasks, ovals to describe conditions and obligations (the word OBL is bold in the graph), double-circles to describe objectives.

The algorithms are able to capture parallelism, in this case between laser and lenses, as well as between glasses, frames and mounting\_machine1; hence, two AND-JOIN gate nodes are present: AND-J_{r,3} is to obtain glasses by using the laser machinery and lenses, AND-J_{r,7} is to obtain eye\_Glasses by using mounting\_machine1, glasses and frames. Notice the node representing mounting\_machine1 has a different notation given that mounting\_machine1 is also an intention in the

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig67.png}
\caption{PeopleEye’s process graph resulting from Algorithm 4 \textsc{complianceBy-Design}: Run of the algorithm up to Line 32.}
\end{figure}
Figure 6.8: PeopleEye’s process graph resulting from Algorithm 4 **complianceByDesign**: Graph with labelled edges.

Figure 6.8 shows the final process graph; ovals for conditions and obligations have been replaced by labels on the corresponding edges.

We now report the theory describing the second running example of Governatore’s Pizza followed by the corresponding process graph.

\[ F = \{ \text{dough, mozzarella, tomato_sauce, cherry_tomatoes, sausage, olives, mushrooms, potatoes, bacon, pesto, Imargherita, Ispring, Idevil Lautumn, Imarianna} \} \]

\[ R = \{ r_1 : \text{dough, mozzarella, tomato_sauce } \Rightarrow \text{margherita} \\
     r_2 : \text{dough, mozzarella, cherry_tomatoes } \Rightarrow \text{spring} \\
     r_3 : \text{dough, mozzarella, tomato_sauce, sausage, olives } \Rightarrow \text{devil} \\
     r_4 : \text{dough, mozzarella, mushrooms, potatoes, bacon } \Rightarrow \text{autumn} \\
     r_5 : \text{dough, mozzarella, pesto, mushrooms } \Rightarrow \text{marianna} \} \]

\[ > = \emptyset. \]

Figure 6.9 shows the process graph of Governatore’s Pizzeria. In this situation, it is clear how the operations of simplification are necessary in order to obtain a graph more structured and easy to understand.

Figure 6.10 depicts the co-occurrence of the tasks DOUGH and MOZZARELLA. Task-literals mozzarella and dough are the raw materials needed for all the pizzas; therefore, it makes sense to link them together with a co-occurrence pattern before making the other simplification operations.

Lastly, Figure 6.11 represents the final process graph. Notice that Algorithm 6 chooses as \( m_s \) at Line 41 tasks MUSHROOMS and TOMATO SAUCE (the
Figure 6.9.: Governatore’s process graph resulting from Algorithm 4 COMPLIANCE-ByDesign: Run of the algorithm up to Line 32.

Figure 6.10.: Governatore’s process graph resulting from Algorithm 4 COMPLIANCE-ByDesign: Co-occurrence of the tasks DOUGH and MOZZARELLA.
6.4. Computational results for synthesis algorithms

We now present the analysis of the computational complexity for the algorithms proposed in this chapter.

Termination as well as computational cost of Algorithm 4 COMPLIANCE-BY-DESIGN depend upon those of its sub-routines, thus we proceed by first proving the two lemmas concerning termination and computational complexity of Algorithm 5 backTrack and of Algorithm 6 splitPATTERN. We remark that the input of Algorithm 4 COMPLIANCE-BY-DESIGN is the positive extension \( E^+(D) \) of a finite defeasible theory \( D \). Therefore, \( E^+(D) \) is finite as well.

**Lemma 29** (Termination and complexity of Algorithm 5). Given a finite, modal defeasible theory \( D \) with size \( \Sigma \) and the corresponding positive extension \( E^+(D) \), Algorithm 5 backTrack terminates and its computational complexity is \( O(\Sigma^2 \cdot \log \Sigma) \).

**Proof.** Algorithm 5 backTrack terminates since literals in \( E^+(D) \) are finite and so are the rules of \( D \), and each rule is considered at most once. Specifically,
whenever the algorithms is invoked, we are in one of the following, mutual
exclusive situations.

1. Literal $l$ has already been analysed: no operations are performed (Line 2).
2. Literal $l \in F$: the corresponding node $L$ is linked with an incoming edge
   from node $\text{START}$ (Line 5).
3. Otherwise, literal $l$ has never been processed and Algorithm 5 $\text{BACKTrack}$
is recursively called for each antecedent $a$. Being the current literal $l$ in
$E^+(D)$, a finite demonstration $P(n)$ proving it exists, and thus eventually
each future recursive call meets either situation 1, or 2.

This bounds also the computational cost. In fact, the only operations which
do not require a constant or linear time are the recursive calls at Lines 11,
20, 31, and 38, which all require $O(\Sigma \cdot \log \Sigma)$. This cost is justified since we
can order elements in those sets: this reduces the time to retrieve elements
in interrogations like the one in Line 11 (an analogous reasoning holds for
operations in Algorithms 4 and 6). Notice that the number of such calls cannot
outnumber the rules involved in the demonstration $P(n)$, which are, in the
worst case, $O(\Sigma)$. We conclude that the complexity is $O(\Sigma^2 \cdot \log \Sigma)$.  

**Observation 30.** Given that each rule is considered only once, and that we did not
bound a literal to differ from any of its antecedent, Algorithm 5 $\text{BACKTrack}$ correctly
manages cycles.

**Lemma 31** (Termination and complexity of Algorithm 6). Given a finite, modal
defeasible theory $D$ with size $\Sigma$, the corresponding positive extension $E^+(D)$, and a
set $r_s \subseteq R$, Algorithms 6 $\text{SPLITPattern}$ terminates and its computational complexity
is $O(\Sigma^4 \cdot \log \Sigma)$.

**Proof.** We use set $r_s \subseteq R$ to denote those rules whose $A(r)$ is in $sL$. At a first, rough
analysis, Algorithm 6 $\text{SPLITPattern}$ terminates since, every time it is recursively
invoked, one (or both) of the following statements holds: (1) Set $sL\text{rest}$ contains
less elements than the corresponding set $sL$, and (2) Some elements $A(r)'$ of
$sL\text{rest}$ contain less literals than the corresponding $A(r)$ in $sL$.

Let us formally justify the previous two statements. When Algorithm 6
$\text{SPLITPattern}$ is invoked, the pivot literal is $l$. Hence, given two elements of $sL$,
then either their intersection is just $l$, or their intersection contains $l$ and at least
another element, namely $a$ in the algorithm. Consequently, we can partition
sets in $sL$ in two parts by the property of containing $a$, or not. The for cycle at
Lines 12-45 chooses co-pivot literals among elements of candidates which, due
to Line 8, does not contain any longer $l$. This justifies statement (2).
Once set $S$ has been chosen, we remove from $sL$ each element containing $S$ (Line 21). Thus, we proved that the next iteration of the cycle for of Lines 12-45, set $sL$ will strictly contain less elements. Moreover, if Algorithm 6 splitPattern is recursively invoked at Line 42, it takes as input set $sLrest$ which, due to Lines 20, 30 and 35, contains less elements than $sL$. These two observations prove statement (1).

Finally, elements of $sLrest$ does not contain any literal in $S$ due to assignment at Line 40. This justifies statement (2) for the recursive call on literal $m$, which is chosen among literals in $sLrest$.

The computational cost of the algorithm is bound to the maximum number of possible recursive calls. It is straightforward by the termination proof to set this number to $O(|\text{elem}_{sL}|) \subseteq O(|\Sigma|)$. It only remains to detail the cost of the other operations. Lines 2, 9-11, 13, 15-19, 22-26, and 33 require constant time. Lines 21, 35, 37 and 44 require linear time. Lines 3, 8 and 40 work in $O(|\Sigma \ast \log |\Sigma|)$, whereas Lines 14, 20 and 30 require quadratic time. The for cycle at Lines 4-7 as well as the one at Lines 31-34 work in $O(|\Sigma| \ast \log |\Sigma|)$. The for cycle at Lines 27-29 works in $O(|\Sigma| \ast \log |\Sigma|)$.

We conclude that the complexity of the for cycle at Lines 12-45 is $O(|\Sigma| \ast \log |\Sigma|)$ times the number of literals in candidates (which is $O(|\Sigma|)$) times the number of admissible recursive iterations of the algorithm, which we proved to be in $O(|\Sigma|)$. This lastly proves our claim. □

We shall now put forward the final result regarding the computational complexity of this doctoral dissertation.

**Theorem 32.** Given a finite, modal defeasible theory $D$ with size $|\Sigma|$ and the corresponding positive extension $E^+(D)$, Algorithm 4 complianceByDesign terminates and its computational complexity is $O(|\Sigma^5 \ast \log |\Sigma|)$.

**Proof.** Operations at Lines 1-3 are in $O(|\Sigma \ast \log |\Sigma|)$.

The for cycle at Lines 10-14 performs at most $O(|\Sigma|)$ iterations. The for cycle at Lines 15-17 performs $O(|\Sigma|)$ iterations times the complexity of Algorithm 5 backTrack, which we proved to be $O(|\Sigma| \ast \log |\Sigma|)$, leading to a cost of $O(|\Sigma| \ast \log |\Sigma|)$. The for cycle at Lines 5-23 is iterated at most $O(|\Sigma|)$ times. Each of its operations requires constant time, with the exception of assignment at Line 6 which requires $O(|\Sigma \ast \log |\Sigma|)$. This binds the cost of the outmost for cycle to $O(|\Sigma^4 \ast \log |\Sigma|)$.

The for cycle at Lines 25-32 is iterated $O(|\Sigma|)$ times. Given that each of its operation requires at most $O(|\Sigma \ast \log |\Sigma|)$, its overall cost is in $O(|\Sigma^2 \ast \log |\Sigma|)$.
The for cycle at Lines 34-40 is iterated $\Sigma$ time and its complexity is bound to that of Algorithm 6 splitPattern, which we demonstrated to be in $O(\Sigma^4 * \log \Sigma)$. Thus, the overall complexity is in $O(\Sigma^5 * \log \Sigma)$.

Lastly, the while cycle at Lines 41-45 is repeated at most $O(\Sigma)$ times. The guard costs $O(\Sigma)$, while operation at Line 42 is in $O(\Sigma^3)$. This bind the complexity to $O(\Sigma^4)$.

Thus, we have $O(\Sigma * \log \Sigma + \Sigma^4 * \log \Sigma + \Sigma^2 * \log \Sigma + \Sigma^5 * \log \Sigma + \Sigma^4)$, for a resulting polynomial complexity. *Quod erat demonstrandum.*

We end this chapter by discussing the correctness results for the last three algorithms. We first need to define what does it mean to be sound and to be complete with respect to process synthesis algorithms.

We start with the concept of completeness. With respect to Algorithm 4 complianceByDesign, being complete means to compute every possible graph given an input logic. Concerning Algorithm 6 splitPattern, being complete means to compute every possible permutation on the pivot literal $l$ (and then $m$). It is trivial to prove that both problems are NP-complete. Under this, perspective, our algorithms are not complete, but neither they were meant to be whereas their purpose was to construct compliant business process graphs (possibly in an efficient way).

We define that the synthesis algorithms are sound if they satisfy the following two properties: (i) All the traces generated by the algorithms are compliant, and (ii) The operations preserve the structure of the theory, i.e., (a) The generated nodes respect sequentiality constraints, that is if rule $r$ is of the form $a \Rightarrow b$, then node $B$ does not occur before node $A$ in the graph, unless there exists $s \in R$ such that $s$ is of the form $A(s) \Rightarrow a$ and $b \in A(s)$; (b) If $r$ is of the form $A(r) \Rightarrow b$ then for each $a_i \in A(r)$, $A_i$ is linked to $B$ through (at least) one AND gate-node; (c) If $b \in C(r)$ and $b \in C(s)$, then node $B$ is linked to each $A_i$, with $a_i \in A(r) \cup A(s)$ through an OR gate-node. All literals considered in point (a), (b), and (c) are in the positive extension of the theory.

**Theorem 33.** Algorithm 4 complianceByDesign is sound.

*Proof.* With respect to the previous definition of soundness, property (i) is guaranteed since $R_{ACT}$ contains only (not defeated) rules with both antecedents and conclusion(s) in the positive extensions of theory $D$. The computation of Algorithms 4–6 starts if, and only if, the theory $D$ given as input of Algorithm 1 DefeasibleExtension is compliant; moreover, $D$ is consistent and coherent, hence if node $A$ is in the graph, then node $\neg A$ cannot be present. Finally, all the “compliant objectives” are met given that the process graph itself is built starting
from such tasks and then a backwards procedure is invoked to the antecedents of the involved rules until the facts are reached.

Property (ii) is ensured by Algorithm 5 \texttt{backTrack} which, given a rule \( r : a_1, \ldots, a_n \rightarrow b \), establishes an equivalence between (1) The causality among \( a_1, \ldots, a_n \) and \( b \), and (2) The sequentiality of nodes representing \( a_1, \ldots, a_n \) with respect to \( b \) (property (a)). Algorithm 6 preserves these properties. Specifically, causality and properties (b) and (c) are guaranteed for the following motivations.

If \( |A(r)| \geq 2 \), then either \texttt{if} condition at Line 14 or \texttt{else} condition at Line 32 of Algorithm 5 \texttt{backTrack} is met and, consequently, the corresponding \texttt{And-J} node is added in between node \( B \) and nodes for literals in \( A(r) \).

If \( |R_{bt}^r| > 1 \) the \texttt{if} condition at Line 24 of Algorithm 5 \texttt{backTrack} is met and, consequently, the corresponding \texttt{Or-J} node is added as the gate which every incoming connection to \( B \) must pass through.

Lastly, the \texttt{And-J} and \texttt{Or-S} gates generated at Line 22 of Algorithm 6 \texttt{splitPattern} are always placed between the elements in \( S \) and the nodes representing conclusions of rules where such elements appear as antecedents. \textit{Quod erat demonstrandum.}

\section*{6.5. Summary and related work}

The contribution of this chapter was twofold: (i) We proposed a methodology to build a process graph starting from the declarative specifications describing capabilities of an agent, the environment she is situated in, and the objectives she is aiming at, and (ii) We advanced a methodology to synthesise such a graph in order to give it a “structure” by incorporating workflow patterns within it. We ended by proving the soundness of the resulting framework, along with its computational feasibility.

There is still much work to do, and many directions which this research may lead to. First, we decided to annotate the edges whereas obligations or literals for conditions are found. An alternative would be to consider such an obligation/condition as a new objective the agent must achieve, stack it in a pile of similar elements and, once the process has completed, invoke Algorithm 4 \texttt{complianceByDesign} on each element of this pile. In this way, we are creating process sub-graphs within the main process graph. Second, we can use this idea to work the other way around. That is, we can recognise sub-graphs that repeat themselves within the main process graph, cut them and substitute them with a single node linked to such a sub-graph. This would unravel the structure of the
main graph.

The approach we presented departs from the standard BDI architecture and agent programming languages implementing it (e.g., 3APL-2APL [Dastani et al., 2005b, Dastani, 2008], JASON [Bordini and Hübner, 2005]), and extensions with norms in several respects (e.g., BOID by Broersen et al. [2002], while we refer the reader to [Alechina et al., 2013] for an overview).

First, the use of reparation chains allows us to exploit the same deductive process to compute what are the norms in force, the outcomes the agent subscribe to, and whether agents are outcome and norm compliant.

Second, while in the above mentioned approaches the agent has to select (partially) predefined plans from a plan library, we propose that the agent generates on the fly a business process (corresponding to a set of plans) to meet the objectives without violating the norms she is subject to.

Finally, 3APL does not make use of declarative specifications attached to each intention. Consequently, intentions are not dropped even when their initial motivating goal is reached. Moreover, in their framework there is no failure condition attached to intentions/goals. Concerning JASON, the use on internal actions is convenient from an implementation perspective but results in blurring the semantic specification.

There are agent frameworks where agents generate plans (e.g., KPG [Kakas et al., 2004] and Golog [Gabaldon, 2011]), but these are typically based on classical AI planning and they do not consider norms and their interactions with other mental attitudes.

Alechina et al. [2012] present a BDI-based agent programming language based on 2APL for norm-aware agents; a norm-aware agent can deliberate on its goals, norms, and sanctions before deciding which plan to select and execute; “a norm-aware agent is able to violate norms (accepting the resulting sanctions) if it is in the agent’s overall interests to do so, e.g., if meeting an obligation would result in an important goal of the agent becoming unachievable”. In this respect, our agents are de facto norm-aware: By distinguishing intentions from social intentions the agent can decide whether to commit to an objective in violation with a norm. Interesting for future lines of research is to develop metrics and evaluation cost functions in such occasions to help the agent in deciding whether to be compliant, or not. A strong point in their work is the assumption that even if the severity of sanctions can be compared, their values are not necessarily commensurable (again for certain people having a criminal record is “less acceptable” than paying a severe fine). On the contrary, a major
issue is that if a goal triggers two (or more) sanctions, each of which is lower in rank than the achievement of that goal, the agent will try to achieve that particular goal even if the sum of the two sanctions is higher in rank. In our opinion, these two issues strongly impact on the soundness of their definition of optimal plan (a plan with maximal utility).

Another interesting work on the same line is that of [Oren et al., 2011], where the authors’ main contribution lies in specifying how an agent should execute a plan, while deciding which norms to adhere to in order to maximise its utility. Along with the formal specifications, the authors advance an algorithm to find optimal plans. The authors describe a technique for taking norms into account when an agent must decide how to execute a plan, reasoning about interactions of norms, and resolve conflicts by choosing set of actions that outweigh the cost of violating certain norms. Their main motivation being that the agent designer, at design time, cannot possibly account for all possible restrictions. As we argued in Chapter 1, these measures are called corrective and are not needed if compliance is taken into account at design time. One of the problems in their approach is that, in general, a designer has adequate knowledge regarding policies and norms governing the environment the agent is acting in, and thus can describe and embed in the framework such obligations at design time.

We spot some other drawbacks in their approach. First, they claim that their notion of obligation is different from works such as [Broersen et al., 2001, Governatori et al., 2007]. $O \varphi \cdot \Gamma$ is not intended with the meaning that $\varphi$ should be executed, but that if $\varphi$ is executed, then it should be done in a way consistent with constraints $\Gamma$. In our framework as in [Governatori et al., 2007], that concept is easily captured by an obligation rule of the form $\varphi \Rightarrow O \Gamma$.

Second, as we proved in [Governatori et al., 2013a], defining the notion of permission as the dual of obligation is an oversimplification within legal systems. Furthermore, the authors use permissions to mitigate violations of norms: This mechanism is conceptually weaker than our reparative chains (as well as their implementation).

Finally, they developed two distinct utility functions to describe gains in achieving a goal and costs of violating norms. It would be interesting for future works to adapt some of their methodologies to develop metrics to evaluate when it is convenient to violate a norm by comparing costs that the agent incurs in violating it against rewards to obtain a critical objective. Two problematics need

---

3Naturally, whenever $\Gamma$ is a set containing $i$ elements, we create an obligation rule for each element in $\Gamma$. 
to be taken into consideration. The former is that their algorithmic solution works in exponential time in the worst case. The latter being that the conceptual work is not as simple as it might appear. The works in the current literature we reviewed are aligned with the solutions proposed in [Oren et al., 2011] (as we have already discussed in Section 4.5) but none considers that, in many occasions, the loss deriving from violating an obligation is “not so easy quantifiable”. When only mere numbers are involved, the matter is not that difficult. If $X$ is the gain of doing action $A$ which violates obligation $O$ with a corresponding fine $Y$, it is convenient to violate $O$ whenever $X > Y$. But what if the violation of $O$ corresponds to a period of detention in jail (with loss of public prestige)? For these reasons, we believe that a more in depth investigation is needed.

Sardiña and Padgham [2011] provide an account of goals in the view of declarative aspects; they describe a goal-failure recovery mechanism able to capture failure handling. Their work is based on previously works: The authors integrate BDI failure mechanisms with HTN planning techniques and they describe the proposed framework as flexible and robust; the agent can choose different plans at various stages of execution and the system tries all applicable plan options to achieve unresolved events. This is also their main drawback: HTN planning is notoriously undecidable even if no variables are allowed, or PSPACE-hard if restrictions either on non-primitive tasks or on the ordering of tasks are given. Similarly to our framework, a belief base is used to describe the agent’s perception of the environment, a transition relation along with derivation rules specify transitions from state to state. The main feature of their CAN is its detailed operational semantics where, if a plan fails, alternative plans for achieving the goal are tried, if possible. Compared to theirs, our framework has the advantage that we generate all the possible plans at design time; even if a plan fails because the environment has changed (due to either new facts or new norms), our algorithms are rather efficient when requested to compute a whole new theory.

Another dissimilarity between our and their framework being in the notion of declarative goal, where a goal must be (i) Persistent, (ii) Unachieved, and (iii) Possible. Nonetheless, notice that the outcomes selected by our process graph algorithms satisfy, by their own nature, such conditions being in the positive extension of the theory. The authors implement a lookahead deliberation, since “the effects of one choice (...) over another is clearly desirable, or even mandatory in order to avoid undesired situations, (...) or again when important resources may be used”. In general, we refer also to [Hindriks et al., 2009], our process
graph algorithms implicitly implement a lookahead mechanism given that their input is the positive extension of a norm and outcome compliant theory. The second part of that work shares little with ours: Given the authors’ focus on recovery techniques (intentions as well as goals are already set) that should be compared to an approach of process revision under compliance.

Using the same goal as ours, Panagiotidi and Vázquez-Salceda [2012] “force” the notion of obligation within the STRIPS framework for agent planning, where the agent is forced to follow some paths, or to avoid some others. As such, their framework is lacking in at least two aspects if compared to ours: (i) They cannot specify the motivational aspects of BDI agents (even if they claim the contrary), (ii) The resulting framework is just another way to tackle agent planning without give an effective process graph as we do. Partially similar is [Panagiotidi et al., 2014], where instead the authors focus more on the life-cycle of norms. Even in that case, their formalisation of violation and timeout is rather convoluted and, to the best of our understanding, quite inefficient.

As already argued in Chapter 1 other works on business processes [van der Aalst and Pesic, 2006, Chesani et al., 2008] use logic tools lured by the fresh perspectives that declarative specifications seem to promise. For our research purposes, both approaches suffer from a great limitation: They capture but constraints between tasks without describing what happens “inside” a task (our effects). Moreover, the logic they use does not allow to handle obligations [van der Aalst and Pesic, 2006], or the notion of obligation is managed but it is a semantically poor concept of obligation since they cannot handle compensations of violated norms [Chesani et al., 2008]. At last, our framework can deal with the notion of preference among goals through the CTD reparative chain mechanism presented in Section 4.1.

Governatori and Rotolo [2010b,a] extend the modal Defeasible Logic of violations introducing three other types of obligations, those being achievement, maintenance and punctual. For an achievement obligation, a certain condition must occur at least once before the deadline; for instance ‘Customers must pay before the delivery of the good, after receiving the invoice’. For maintenance obligations, a certain condition must obtain during all instants before the deadline; for example, ‘After opening a bank account, customers must keep a positive balance until bank charges are taken out’. Lastly, punctual obligations apply only to single instants or tasks; mathematically they can be thought as either maintenance obligations or achievement obligations in force in time intervals where the endpoints are equal. Typically, punctual obligations must occur at
the same time of their triggering conditions; for example, ‘Customers must pay before the delivery of the good, after receiving the invoice. Otherwise, an additional fine must be paid’, or ‘After opening a bank account, customers must keep a positive balance until bank charges are taken out. Otherwise, their account is blocked’. In those works, exploiting the mechanism of reparations chain extended with many type of obligations, the authors capture AND, OR and eXclusive OR patterns.

In [van der Aalst and Pesic, 2006], Linear Temporal Logic operators\(^6\) are used to create constraint templates, consisting in a formula and its graphical representation: their purpose is to control the service flow. We want to combine ideas expressed by the constraint templates with the many type of obligations of [Governatori and Rotolo, 2010b,a] to extend our logic. In this way, we might improve how to handle parallel and choice patterns as well temporal and flow constraints.

Backward graph approach was not the only approach we examined as a valid solution for the problem of compliance by design. Our first steps were inspired by the works of van Dongen et al. [2008], Dijkman et al. [2011] and moved in the direction of studying a measure of distance among processes, i.e., to define a notion of similarity between tasks. Our intuition was that, given a pool of business processes “similar” to the process we want to obtain, similarity would allow us to assemble many parts together in order to obtain the final process.

As far as this approach seemed promising, from its incipit it showed some strong limitations. To begin with, even if we can collect a repository of similar tasks, the process of deciding if a task is similar to a second one (whatever the intended meaning of similarity could be, e.g., same input/output, same resources needed, etc.) says nothing about compliance. Moreover, a non trivial question is where this pool of similar tasks comes out.

Secondly, as we argued in the introduction, the information about the organisation at hand is not contextualised in fully formalised processes but in the form of repository of capabilities. In this scenario, similarity can handle only local information, telling if a fragment can be replaced with another. Yet, the final aim is to obtain an entire business process and similarity is not able to assemble all these pieces of information together.

Finally, the more specific cumbersome aspect of similarity is that, when we look at a process from the purely structural viewpoint, if the input and the

\(^6\)Temporal operators could be: next-time ($\bigcirc F$), eventually ($\Diamond F$), always ($\Box F$) and until ($F \sqcup G$).
output of a fragment are compatible, we can replace the sub-process with the new one. Again, this is but a local operation. When we consider compliance aspects, the semantic interactions we have to deal with (i.e., the effects of all the activities inside a single fragment) must be analysed globally, on the entire process, and not just in a local context. We must consider the interactions with all the activities of the other subprocesses.

Motivated by these considerations, we decided to abandon the path of similarity.

In the current literature, another approach appears promising for realising compliance by design and it consists in applying techniques based on process mining research to induce a process graph starting from workflow logs to the many derivations from the theory leading to goals and norms.

Process mining approach. Agrawal et al. [1998], van der Aalst et al. [2004], define methodologies and algorithms for discovering workflow models starting from workflow logs. A workflow log contains informations about the workflow as it is actually being executed: all the traces in a workflow log are a representative and sufficiently large subset of the possible behaviours of systems modelled in the workflows themselves. Through process mining, the authors start from linear sequences of tasks to obtain complex structures able to capture patterns like parallelism and choices.

We can apply the same methodologies in our case. The statement is motivated by the following reasoning. Given a reachable goal, a derivation for it is a linear sequence of (proved) literals in the theory. Thus, such a derivation can be understood as a log trace. Even if, to obtain a literal, the derivation rule has a set of premises that contains more than a single element, Antoniou et al. [2008] provides procedures to automatically obtain derivations. For example, given the theory with the only rule \( r : p, q \Rightarrow t \), where \( p \) and \( q \) are facts, we obtain as derivations \( p \rightarrow q \rightarrow t \) and \( q \rightarrow p \rightarrow t \).

Since we proved the initial theory to be compliant, each derivation corresponds to a compliant trace. The issue is how to extract such derivations and combine the corresponding traces to form a single process graph. The main challenge in inducing such a graph from a log of past executions is to identify dependencies between activities. While in data mining approach, process logs must be analysed in order to identify the sequence classes of relevant executions, in our logic each derivation is representative since a derivation is a compliant execution of the

\(^{7}\)Evidently, only a part of the regularities in these traces can be used by such a technique, and therefore the authors recommend to apply their techniques in presence of a large data set.
Even if our current research was mainly focused on the backward graph approach, it might appear of a certain importance to deepen the investigation in the process mining approach as well. The reason is twofold: (i) Unlike similarity of processes, mining approach seems showing great potential, (ii) A second approach would allow not just a mere alternative to the backward graph approach but a valid tool to compare and evaluate performances of both approaches.

The last branch of literature we looked at to improve our techniques was that of Service Composition, which is the branch of literature studying how to “put together” a selection of services chosen among a larger pool in order “to create, using a subset of already available Web services as atomic building blocks, a more complex Web service providing a more useful functionality” [Heymans et al., 2011]. A service is defined by its interface that specifying activation conditions (premises) as well as which effects the execution of such a service has brought about. Service Composition uses planning techniques to combine services. Notice that planning is computationally hard in general, but AI research has come up with a range of rather effective approaches, e.g., [Hoffmann and Nebel, 2001, Richter and Westphal, 2010].

Given that there is no guarantee that plans are effectively executable, it is typical that the planning facility accomplishes only a part of the service composition design and that a human user is responsible for the overall design, and uses the planning facility for easing the task of selecting and arranging a useful subset of services from a large services database or the internet (e.g., [Agarwal et al., 2005, Hoffmann et al., 2010, Bertoli et al., 2010]).

In order to run the planning facility, services need to described as actions in the first place, and the user needs to enter the initial state and goal of the desired composed service. These are non-trivial tasks. Several frameworks have appeared that allow users to conveniently design the models via user-friendly interfaces, e.g., [Rodríguez-Moreno et al., 2007, Celorio et al., 2013, Gonzales-Ferrer et al., 2009]. An alternative approach suggests to instead exploit pre-existing models of software behaviour [Hoffmann et al., 2010].

Very close to our approach and methodology is [Helmert, 2009], where the author defines a plan as the path in the state space leading from the initial state to a set of states containing the so-called goal state. To the best of our knowledge,

---

8I personally thank Heymans et al. [2011] given that many of the following contents were taken from their most useful summary in Chapter 6 of Handbook of Service Description: USDL and its Methods.
our work goes further than the ones of that sector in that we synthesise the graph through the final three algorithms proposed in the present chapter. Moreover, and by far not the least important of issues, is the problem of deciding whether or not there exists a plan (with their formalism) is computationally hard (PSPACE-complete).

In ASTRO framework [Marconi et al., 2006, Bertoli et al., 2009] the basic idea is that existing services can be used to construct the planning domain, composition requirements can be formalised as planning goals, and planning algorithms can be used to generate plans that compose the published services. The ASTRO framework has been widely adopted to deal with the different aspects of the Web service composition problem. In particular, ASTRO has been enabled to specify complex data-flow [Marconi et al., 2006] and control-flow composition requirements [Bertoli et al., 2009] and an abstraction-based approach for composing services that manipulate complex, infinite-range data domains [Pistore et al., 2005].

Lastly, we reviewed Description Logic applied to planning [Lifschitz, 2002]. Description Logic makes use of interpretation where we use extensions $\pm\Delta$, and $\pm\delta$). One of the main issues here is that Description Logic contains non-trivial axiomatisations of the domain. These axioms make it difficult to define and compute the “outcome state” resulting from applying an action.

To the best of our understanding, all these approaches using planning techniques suffer a main drawback: they work with constraints, whereas our logic uses (reparative chains of) obligations. The problem being when a constraint is violated, no reparation is allowed and the whole plan becomes non-executable. In our logic, instead, when an obligation occurs, the plan can still be valid by considering compensations.
CHAPTER
SEVEN

THERE AND BACK AGAIN: CONCLUSIONS AND FAREWELL

There are more things
in heaven and earth, Horatio,
Than are dreamt of in your philosophy.

Hamlet (1.5.167-8) - William Shakespeare

The goal of this doctoral dissertation was to propose a framework for obtaining structured business processes starting from the declarative specifications describing means of an organisation, its business objectives, as well as the normative aspects governing the working environment.

We began by introducing the concept of business processes as tools to describe and optimise activities of an organisation, and we continued by describing the approaches present in the literature of the field, categorising them in two families, imperative and declarative approaches. We then motivated our choice to set the framework within the declarative family, and explained why a logic approach may be one of the best candidates. We linked logics to graphs, and introduced the non-monotonic, skeptical logic underlying our framework, as well as the graphical notation adopted.

We then moved to describe, and motivate, all reasons behind the modal Defeasible Logic we proposed thereafter. We concluded the presentation of our logic by proposing algorithms apt to compute the sets of positive and negative conclusions given a particular set of facts, proving their correctness as well as their computational tractability.

Lastly, we “transformed” all such logical derivations into traces of a process
graph. The graph is constructed through a backwards approach: The algorithms start by identifying the achievable objectives and recursively navigate backwards exploring rule by rule, antecedent after antecedent. After this initial phase, we “give a structure” to the graph by recognising OR and AND gateways. Again, we ended by proving that the proposed methodologies are sound and computationally efficient. These results combined with the results on the logic prove that the whole framework proposed in this Ph.D. dissertation works in polynomial time.

Finally, Appendix A reports complete proofs of lemmas and theorems proving the correctness of Algorithms 1 to 3; Appendix B reports a detailed run of Algorithm 6 splitPattern on a toy theory. The two running Examples 9 and 10 show how efficacious and efficient our methods are in representing two real life scenarios, one being that of an eye glasses manufacturer, the other of a pizzeria.

Even if the main topic of the thesis is business process compliance, we firmly believe that this doctoral dissertation reached compelling topics in the cross areas of knowledge representation, agent theory, plan recognition, and graph theory. Our logical framework is not just fitted to describe real life problems but, despite most of the works nowadays present in the literature [Dastani et al., 2008, Hindriks and van Riemsdijk, 2008, Vasconcelos et al., 2009, Helmert, 2009, Grant et al., 2010a, Oren et al., 2011, Sardiña and Padgham, 2011, Shapiro et al., 2012], our approach is computationally efficient. Our logic is suited to describe business processes by adopting a declarative approach, and properly represents an agent deliberation process.

We proposed a fresh characterisation for the concepts of desires, goals, intentions, and social intentions which are motivational states obtained through a deliberative process based on various types of preferences among desired outcomes. Our intuition was aligned with the BDI philosophy according to which intentions and goals are the results of the interaction between the facts describing the environment (in the form of beliefs) and the agent’s desires. Moreover, we used ideas present in the BOID architecture and in the BIO logics [Broersen et al., 2002, Governatori and Rotolo, 2008b] to filter objectives through norms and to solve conflicts among different rule types.

We distinguished concepts of desire, goal, intention and social intention, but we started from the shared notion of outcome. Therefore, they are a single notion that becomes distinct based on the particular relationship with beliefs and norms. This reflects a more natural notion of mental attitude and can express the well-known notion of Plan B. When we consider the single chain itself, this
justifies that from a single concept of outcome we can derive all the other mental attitudes. Otherwise we would need as many additional rules as elements in the chain; this, in turn, would require the introduction of additional notions to establish the relationships with beliefs and norms. This adds to our framework an economy of concepts, and as Okkham said, “entia non sunt multiplicanda praeter necessitatem”.

Moreover, since the preferences allow us to determine what preferred outcomes are adopted by an agent (in a specific scenario) when previous elements in sequences are not (or no longer) feasible, our logic provides an abstract semantics for several types of goal and intention reconsideration.

Finally, our backwards approach put forward methodologies on how to create a graph starting from a logical description; our synthesis methods showed how to recognise parallel and choice patterns, and introduced interesting ways on how to group nodes and simplify the graph itself.

We pointed out the strengths in our approach, and now we shall discuss some drawbacks. First and foremost, a question may arise: Are our methods directly implementable or usable by a person without a Ph.D. in computer science? Regarding the first part, the answer is no, while for the second part of our approach the answer is yes (graph representation is one of the most used standard in business process). The problem lies in the difficulty of translating a natural language description into a logic formalisation. This is a notoriously hard task.

Even if the obstacle seems very difficult, the payoff is worthwhile. The first reason is due to the efficiency of the computation of the positive extension once the formalisation has been done (polynomial time against the majority of the current frameworks in the literature which typically work in exponential time). The second reason is that the use of business rules to describe complex systems is extremely common [Knolmayer et al., 2000]. Future lines of research will then focus on developing such methods, by giving tools which may help the business analyst in writing such business rules from the declarative description.

Another issue comes from the fact that, typically, systems implemented by business rules involve thousands of such rules. Again, our choice of Defeasible Logic allows to drastically reduce the number of rules involved in the process of creating a business process thanks to its exception handling mechanism. This is peculiarly interesting when dealing with the problem of visualising such rules. When dealing with a system with thousands of rules, understanding what they represent or what a group of rules stand for, may be a serious challenge. On
the contrary, our model, once the input is given, allows for the identification of whether the whole process is compliant against a normative system and a set of goals (and if not, where it fails). To our best knowledge, no other system is capable of checking whether a process can start with its input requisites and reaches its final objectives in a way that is compliant with a given set of norms.

The present framework is meant to be seen as the first step within a more general perspective of providing the business analyst with tools that allow the creation of a business process in a fully declarative manner (as the trend of [Agrawal et al., 1998, Pesic et al., 2007, Chesani et al., 2008]). But, in contrast of such frameworks, our approach has many strong points. It is more flexible due to the fact that we have an (efficient) exception handling mechanism capture. Moreover, by taking into consideration the many derivations to prove one literal and by translating them into the graph, you actually have more ways to reach the node representing that specific literal. Notice that we may well consider flexibility the reason that our algorithms work altogether in polynomial time: This implies that when some facts in the environment change, a new graph can be re-computed in reasonable time. At last, we have already argued the importance of a norm impacting the organisational environment; therefore, such a concept cannot be overlooked, or treated in a simplistic manner. Our logics captures the concept of norms. It is true that a mathematical formalisation might be hard to grasp for a non-expert of the field, but it gives accuracy to the whole framework given that it allows to prove the correctness of the formalisation, as we have done. Regardless, the complexity of the mathematical formalisation does not imply that the approach itself is hard to understand: Once formalised, the meaning of the rules is easy to explain to an end-user. This was not the final aim of this doctoral dissertation.

We have already pointed out some future lines of research that may follow the theoretical and practical work behind this thesis. And as the ancient Romans used to say: *repetita iuvant.*

The logical apparatus will be extended to include different types of obligations and goals, as maintenance and achievement [Governatori and Rotolo, 2010b,a, Dastani et al., 2006, Hindriks and van Riemsdijk, 2007, 2008], as well as strong and weak permissions as presented in [Governatori et al., 2013a]. Temporal constraints need to be incorporated as well; explicit time will close the gap between our formalisation and how norms are expressed in real life normative systems even more, given that key components of a norm are its validity period and deadlines. It will be interesting how evaluation measures can be introduced
within the logics, given that both norms and goals often are expressed by means of thresholds (for example, during a refrigeration process the meat cannot stay at a temperature higher of -20°C for more than 2 minutes, otherwise it is no longer considered safe for human consumption). Clearly, the use of variables in agent programming languages is of the utmost importance. Anyhow, this poses several difficulties, resulting in a framework with a notation considerably more complicated. From an abstract point of view, we will need to take into account several aspects, such as (i) They need to be propagated during their execution, (ii) When the framework will consider explicit time, some strategies may be applicable for different installations of such variables, (iii) Shared variable are delicate to handle in parallel execution. Finally, we want to explore in more detail and improve the cost and value functions proposed in [Hindriks and van Riemsdijk, 2008, Grant et al., 2010a]. In Section 4.5, we have already discussed how the ideas presented in those works are interesting but lack in many aspects. We need to understand how to integrate a cost and value function within our framework that can assist the agent in non-trivial choices (for example, when the loss of violating an obligation is not directly quantifiable). This will result in a consistent empowerment of our logics.

Declarative-type approaches have the advantage that it is usually simple enough to develop mapping tools. As said before, a powerful tool is to develop a software to map a natural language description into a semi-logical language; such a tool would help the business analyst through forms, for example lists of machineries present on the market for that particular type of business, lists of human resources and roles, lists of financial resources, lists (more or less complete) of norms through queries of normative thesauri. Such a tool will “recognise” the pertinent business area, and help the business analyst in describing resources available and thus defining the business objectives. This process will assist the business analyst in identifying existing and new tasks, as well as in discovering whether existing tasks may be used in a non-conventional manner (for instance, it has been recently discovered that it is feasible and profitable to use settling tanks to produce ammonium salt, where they are typically used to produce acids). In this manner, the description of the resources is the description of the tasks associated to such resources. We believe that this approach may be helpful also in a second phase when the business analyst, looking at the process graph, can revise it, build a new process graph, and start the procedure again until a desirable process is obtained. Related to this first task, is also the translation of a normative system into a defeasible deontic logic.
At first glance, it might appear that our framework does not make any type of optimisation of the final business process. On the contrary, our optimisation process lies in how we obtain the logic given a particular declarative description. In fact, with a different description we obtain a different logic and, consequently, a different process graph; a business analyst can compare the two different solutions. This is what we understand for optimisation, given that we believe that an efficient optimisation process cannot be done before the generation of the business graph, but only after that phase when an expert can look at the resulting graph and decide whether it matches the expectations, or needs, to be changed. This is in line with the business practices where feedbacks by the front-end users, employees and actors involved in the production process are used ex-post to improve an already existing business process.

The work we have done might be classified as plan recognition, which is inferring models of plans by the agent’s observation of actions and goals expected to be pursued. An inference system has the task of inferring the agent’s goals and determining how the observed actions contribute to that goal [Carberry, 2001]. To accomplish this, the system is traditionally provided with a set of actions that can be executed by the agent and a set of recipes describing each action’s preconditions and effects. Ultimately, the system will chain actions to goals, then these goals to other actions, and those actions to their goals, et cetera. Even if the literature has just started to be reviewed, we think that in future works we can use plan recognition techniques in developing the tools we need to identify plans and goals (and thus subsequently create the corresponding theory) from the natural language.

The journey which I started four long years ago in the beautiful country of Australia is finally coming to an end. What comes next, only God knows. If you, who have read these many pages have made it to these final lines, I thank you for your patience and hope not to have bored you. At least not too much.
We report the lemmas used by Theorem 26 for the soundness and completeness of the algorithms proposed.

The algorithms in Chapter 5.2 are based on a series of transformations that reduce a given theory into an equivalent “simpler” one. Here, equivalent means that the two theories have the same extension, and simpler means that the size of the target theory is smaller than that of the original one. Remember that the size of a theory is the number of instances of literals occurring in the theory plus the number of rules in the theory. Accordingly, each transformation either removes some rules or some literals from rules (specifically rules or literals we know are no longer useful to produce new conclusions). There is an exception. At the begin of the computation, the algorithm creates four rules (one for each type of goal-like modality) for each outcome rule (and the outcome rule is then eliminated). The purpose of this operation is to simply the transformation operations and the bookkeeping of which rules have been used and which rules are still able to produce new conclusions (and the type of conclusions). Alternatively, one could implement flags to achieve the same result, but in a more convoluted way. A consequence of this operation is that we no longer have outcome rules. This implies that we have to (i) adjust the proof theory, (ii) show that the adjusted proof theory and the theory with the various goal-like rules is equivalent to the original theory and original proof conditions.

The adjustment required to handle the replacement of each outcome rule with a set of rules of goal-like modes (where each new rule has the same body and consequent of the outcome rule it replaces) is to modify the definition of
applicable for $X$ (Definition 7) and discarded for $X$ (Definition 8). Specifically, we have to replace

- $r \in R^U$ in clause 3 of Definition 7 with $r \in R^D$;
- $r \not\in R^U$ in clause 3 of Definition 8 with $r \not\in R^D$;
- $r \in R^X$ in clause 4.1.1 of Definition 7 with $r \in R^X$; and
- $r \not\in R^X$ in clause 4.1.1 of Definition 8 with $r \not\in R^X$.

Given a theory $D$ with goal-like rules instead of outcome rules we shall use $E_4(D)$ to refer to the extension of $D$ computed using the proof theory defined in Chapter 4 with the modified versions of the notions of applicable and discarded just given.

**Proposition 34.** Let $D = (F, R, >)$ be a defeasible theory. Let $D' = (F', R', >')$ be the defeasible theory obtained from $D$ as follows:

$$R' = R^B \cup R^O \cup$$

$$\{r_X : A(r) \leftarrow_X C(r) | r : A(r) \leftarrow \cup C(r) \in R, X \in \{D, G, I, SI\}\}$$

$$>'_\cup = \{(r, s) | (r, s) \in >, s, r \in R^B \cup R^O\} \cup$$

$$\{(r_X, s) | (r, s) \in >, r, s \in R^U\}$$

$$\{(r_X, s) | (r, s) \in >, r \in R^U, s \in R^B \cup R^O\} \cup$$

$$\{(r, s_X) | (r, s) \in >, r \in R^B \cup R^O, s \in R^U\}$$

Then, $E(D) = E_4(D')$.

**Proof.** The differences between $D$ and $D'$ are that each outcome-rule in $D$ corresponds to four rules in $D'$ each for a different mode and all with the same antecedent and consequent of the rule in $D$. Moreover, every time a rule $r$ in $D$ is stronger than a rule $s$ in $D$, then any rule corresponding to $r$ in $D'$ are stronger than any rule corresponding to $s$ in $D'$.

The differences in the proof theory for $D$ and that for $D'$ is in the definitions of applicable for $X$ and discarded for $X$. It is immediate to verify that every time a rule $r$ is applicable (at index $n$) for $X$, then $r_X$ is applicable (at index $n$) for $X$ (and the other way around). 

Given the functional nature of the transformations involved in the algorithms, we shall denote the rules in the target theory with the same labels as the rules in the source theory. Thus, given a rule $r \in D$, we shall refer to the rule corresponding to it in $D'$, if it exists, with the same label, namely $r$.

In the algorithms, belief rules may convert to another mode $\diamond$ only through the support set $R^{B,\diamond}$. Definition 5 requires $R^{B,\diamond}$ to be initialised with a modal...
version of each belief rule with *non-empty* antecedent, such that every literal $a$ in the antecedent is replaced by the corresponding modal literal $\Diamond a$.

In this manner, rules in $R^{B,\Diamond}$ satisfy clauses 1 and 2 of Definitions 5 and 6 by construction, while clauses 3 of both definitions are satisfied iff these new rules for $\Diamond$ are body-applicable (resp. body-discarded). Therefore, conditions for rules in $R^{B,\Diamond}$ to be applicable/discarded collapse into those of Definition 3 and 4, and accordingly these rules must satisfy clauses (2.1.1), (3.1), or (4.1.1) of Definitions 7–8 instead of (2.2), (3.2) and (4.2). That is to say, during the execution of the algorithms, we can empty the body of the rules in $R^{B,\Diamond}$ by iteratively proving all the modal literals in the antecedent to decide which rules are applicable at a given step.

Before proceeding with the demonstrations of the lemmas, we recall that in the formalisation of the logic in Chapter 4, we referred to modes with capital roman letters ($X, Y, T$) while the notation of the algorithms in Chapter 5 proposes the variant with $\Box, \blacksquare$ and $\Diamond$ since it was needed to fix a given modality for the iterations and pass the correct input for each call of a subroutine. Therefore, being the hypotheses of the lemmas referring to the operations performed by the algorithms while the proofs refer to the notation of Definitions 3–13, in the following the former ones use symbol $\Box$ for a mode, the latter ones the capital roman letters notation.

**Lemma 35.** Let $D = (F, R, >)$ be a modal defeasible theory such that $D \vdash +\partial_{\Box}l$ and $D' = (F, R', >')$ be the theory obtained from $D$ where

$$R' = \{r : A(r) \setminus \{\Box l, \neg \Box \neg l\} \leftarrow C(r) | r \in R, A(r) \cap \Box \neg l = \emptyset\}$$

$$R^{B,\Diamond} = \{r : A(r) \setminus \{\Box l\} \leftarrow C(r) | r \in R^{B,\Diamond}, A(r) \cap \Box \neg l = \emptyset\}$$

$$> ' = > \setminus \{(r, s), (s, r) \in > | A(r) \cap \Box \neg l = \emptyset\}.$$  

Then $D \equiv D'$.

**Proof.** The proof is by induction on the length of a derivation $P$. For the inductive base, we consider all possible derivations for a literal $q$ in the theory.

$P(1) = +\partial_X q$, with $X \in \text{MOD} \setminus \{D\}$. This is possible in two cases: (1) $Xq \in F$, or (2) $\neg \neg q \not\in F$, for $Y = X$ or Conflict$(Y, X)$, and $\exists r \in R^X[q, i]$ that is applicable in $D$ for $X$ at $i$ and $P(1)$, and every rule $s \in R^Y[\neg q, j]$ is either (a) discarded for $X$ at $j$ and $P(1)$, or (b) defeated by a stronger rule $t \in R^T[q, k]$ applicable for $T$ at $k$ and $P(1)$ ($T$ may conflict with $Y$).

Concerning (1), by construction of $D'$, $Xq \in F$ iff $Xq \in F'$, thus if $+\partial_X q$ is
provable in $D$ then is provable in $D'$, and vice versa.

Regarding (2), again by construction of $D'$, $\overline{Y} q \notin F$ iff $\overline{Y} q \notin F'$. Moreover, $r$ is applicable at $P(1)$ iff $i = 1$ (since lemma’s operations do not modify the tail of the rules) and $A(r) = \emptyset$. Therefore, if $A(r) = \emptyset$ in $D$ then $A(r) = \emptyset$ in $D'$. This means that if a rule is applicable in $D$ at $P(1)$ then is applicable in $D'$ at $P(1)$. In the other direction, if $r$ is applicable in $D'$ at $P(1)$ then either (i) $A(r) = \emptyset$ in $D$, or (ii) $A(r) = \{l\}$, or $A(r) = \{\neg\Box \neg l\}$. For (i), $r$ is straightforwardly applicable in $D$, as well as for (ii) since $D + +\partial_l l$ by hypothesis.

When we consider possible attacks to rule $r$, namely $s \in R^Y[\neg q, j]$, we have to analyse cases (a) and (b) above.

(a) Since we reason about $P(1)$, it must be the case that no such rule $s$ exists in $R$, and thus $s$ cannot be in $R'$ either. In the other direction, the difference between $D$ and $D'$ is that in $R$ we have rules with $\overline{\partial l}$ in the antecedent, and such rules are not in $R'$. Since $D + +\partial_l l$ by hypothesis, all rules in $R$ for which there is no counterpart in $R'$ are discarded in $D$.

(b) We modify the superiority relation by only withdrawing instances where one of the rules is discarded in $D$. But only when $t$ is applicable then is active in the clauses of the proof conditions where the superiority relation is involved, i.e., (2.3.2) of Definition 11. We have just proved that if a rule is applicable in $D$ then is applicable in $D'$ as well, and if is discarded in $D$ then is discarded in $D'$. If $s$ is not discarded in $D$ for $Y$ at 1 and $P(1)$, then there exists an applicable rule $t$ in $D$ for $q$ stronger than $s$. Therefore $t$ is applicable in $D'$ for $T$ and $t > s$ if $T = Y$, or Conflict($T, Y$). Accordingly, $D' + +\partial q$. The same reasoning applies in the other direction. Consequently, if we have a derivation of length 1 of $+\partial q$ in $D'$, then we have a derivation of length 1 of $+\partial q$ in $D$ as well.

Notice that in the inductive base by their own nature rules in $R^B, \cap$, even if can be modified or erased, cannot be used in a proof of length one.

$P(1) = +\partial D q$. The proof is essentially identical to the inductive base for $+\partial X q$, with some slight modifications dictated by the different proof conditions for $+\partial q$: (1) $D q \notin F$, or (2) $\neg D q \notin F$, and $\exists r \in R^D[q, i]$ that is applicable for $D$ at 1 and $P(1)$ and every rule $s \in R^D[\neg q, j]$ is either (a) discarded for $D$ at 1 and $P(1)$, or (b) $s$ is not stronger than $r$.

$P(1) = -\partial X q$ with $X \in \text{MOD}$. Clearly conditions (1) and (2.1) of Definition 12 hold in $D$ iff they do in $D'$, given that $F = F'$. The analysis for clause (2.2) is the same of case (a) of $P(1) = +\partial X q$, while for clause (2.3.1) the reader is
referred to case (2), where in both cases $r$ and $s$ change their role. For condition (2.3.2) if $X = D$, then $s > r$. Otherwise, either there is no $t \in R^T[q,k]$ in $D$ (we recall that at $P(1)$, $t$ cannot be discarded in $D$ because that would imply a previous step in the proof), or $t \not\in s$ and not Conflict($T,Y$). Therefore $s \in R'$ by construction, and conditions on the superiority relation between $s$ and $t$ are preserved. Hence, $D' \vdash -\partial_X q$. For the other direction, we have to consider the case of a rule $s$ in $R$ but not in $R'$. As we have proved above, all rules discarded in $D'$ are discarded in $D$, and all rules in $R$ for which there is no corresponding rule in $R'$ are discarded in $D$ as well, and we can process this case with the same reasoning as above.

For the inductive step, the property equivalence between $D$ and $D'$ is assumed up to the $n$-th step of a generic proof for a given literal $p$.

$$P(n + 1) = +\partial_X q, \text{ with } X \in \text{MOD}. \quad \text{Clauses (1) and (2.1) follow the same conditions treated in the inductive base for } +\partial_X q. \text{ As regards clause (2.2), we distinguish if } X = B, \text{ or not. In the former case, if there exists a rule } r \in R[q,i] \text{ applicable for } B \text{ in } D, \text{ then clauses 1–3. of Definition 3 are all satisfied. By inductive hypothesis, we conclude that the clauses are satisfied by } r \text{ in } D' \text{ as well no matter whether } \overline{a} \in A(r), \text{ or not.}$

Otherwise, there exists a rule $r$ applicable in $D$ for $X$ at $P(n+1)$ such that $r$ is either in $R^X[q,i]$, or $R^{B,X}[q,1]$. By inductive hypothesis, we can conclude that:

(i) If $r \in R^X[q,i]$ then $r$ is body-applicable and the clauses of Definition 3 are satisfied by $r$ in $D'$ as well; (ii) If $r \in R^{B,X}[q,1]$ then $r$ is Conv-applicable and the clauses of Definition 5 are satisfied by $r$ in $D'$ as well. As regards conditions (2.1.2) or (4.1.2), the provability/refutability of the elements in the chain prior to $q$ is given by inductive hypothesis. The direction from rule applicability in $D'$ to rule applicability in $D$ follows the same reasoning and so is straightforward.

Condition (2.3.1) states that every rule $s \in R^Y[\neg q,j] \cup R^{B,Y}[\neg q,1]$ is discarded in $D$ for $X$ at $P(n + 1)$. This means that there exists an $a \in A(s)$ satisfying one of the clauses of Definition 4 if $s \in R^{B,Y}[\neg q,1]$, or Definition 8 if $s \in R^Y[\neg q,j]$. Two possible situations arise. If $a \in \overline{a}$, then $s \not\in R'$; otherwise, by inductive hypothesis, either $a$ satisfies Definition 4 or 6 in $D'$ depending on $s \in R^Y[\neg q,j]$ or $s \in R^{B,Y}[\neg q,1]$. Hence, $s$ is discarded in $D'$ as well. The same reasoning applies for the other direction. The difference between $D$ and $D'$ is that in $R$ we have rules with $\overline{a}$ in the antecedent, and these rules are not in $R'$. Consequently, if $s$ is discarded in $D'$, then is discarded in $D$ and all rules in $R$ for which there is no corresponding rule in $R'$ are discarded in $D$ since $D \vdash +\partial_\overline{a}$ by hypothesis.
If \( X \neq D \), then condition (2.3.2) can be treated as case (b) of the corresponding inductive base except clause (2.3.2.1) where if \( t > s \) then either: (i) \( Y = T \),
(ii) \( s \in R^{B,T}[\neg q] \) and \( t \in R^T[q] \) (Convert\((Y,T)\)), or (iii) \( s \in R^Y[\neg q] \) and \( t \in R^{B,Y}[q] \) (Convert\((T,Y)\)). Instead if \( X = D \), no modifications are needed.

\[ P(n + 1) = -\partial_X q, \text{ with } X \in \text{MOD}. \]
The analysis is a combination of the inductive base for \( -\partial_X q \) and inductive step for \( +\partial_X q \) where we have already proved that a rule is applicable (discarded) in \( D \) iff it is already contained in \( R' \). Even condition (2.3.2.1) is just the strong negation of the reason in the above paragraph.

**Lemma 36.** Let \( D = (F,R,>) \) be a modal defeasible theory such that \( D \vdash -\partial_{\square} l \)
and \( D' = (F,R',>) \) be the theory obtained from \( D \) where

\[
R' = \{ r : A(r) \setminus \{ \neg \square l \} \leftarrow C(r) \mid r \in R, \sqvee l \notin A(r) \}
\]

\[
R^{B,\square} = \{ r \in R^{B,\square} \mid \square l \notin A(r) \}
\]

\[
> ' = > \setminus \{(r,s),(s,r) \in > | \sqvee l \in A(r) \}.
\]

Then \( D \equiv D' \).

**Proof.** We split the proof in two cases, depending on if \( \square \neq D \), or \( \square = D \).

As regards the former case, since Proposition 16 states that \( +\partial_X m \) implies \( -\partial_X \neg m \) then modifications on \( R' \), \( R^{B,\square} \), and \( >' \) represent a particular case of Lemma 35 where \( m = \neg l \).

We now analyse the case when \( \square = D \). The analysis is identical to the one shown for the inductive base of Lemma 35 but for what follows.

\[ P(1) = +\partial_X q. \]

Case (2)–(ii): \( A(r) = \{ \neg \square l \} \) and since \( D \vdash -\partial_{\square} l \) by hypothesis, then if \( r \) is applicable in \( D' \) at \( P(1) \) then is applicable in \( D \) at \( P(1) \) as well.

Case (2)–(a): The difference between \( D \) and \( D' \) is that in \( R \) we have rules with \( \square l \) in the antecedent, and such rules are not in \( R' \). Since \( D \vdash -\partial_{\square} l \) by hypothesis, all rules in \( R \) for which there is no counterpart in \( R' \) are discarded in \( D \).

The same modification happens in the inductive step \( P(n + 1) = +\partial_X q \), where also the sentence ‘If \( a \in \square l \), then \( s \notin R' \)’ becomes ‘If \( a = \square l \), then \( s \notin R'' \).

Finally, the inductive base and inductive step for the negative proof tags are identical to ones of the previous lemma.

Hereafter we consider theories obtained by the transformations of Lemma 35. This means that all applicable rules are such because their antecedents are empty.
and every rule in $R$ appears also in $R'$ and vice versa, and there are no modifications in the antecedent of rules.

**Lemma 37.** Let $D = (F, R, >)$ be a modal defeasible theory such that $D + \Box \alpha l$ and $D' = (F, R', >)$ be the theory obtained from $D$ where

\[
R^O = \{ A(r) \Rightarrow O \ l | r \in R^O[l, n] \} \quad (A.0.1)
\]
\[
R^I = \{ A(r) \Rightarrow I \ l | r \in R^I[l, n] \} \cup \{ A(r) \Rightarrow I \ l | r \in R^I[\sim l, n] \} \quad (A.0.2)
\]
\[
R^{SI} = \{ A(r) \Rightarrow SI \ l | r \in R^{SI}[\sim l, n] \} \quad (A.0.3)
\]

Moreover,

- if $D + \Box O \sim l$, then instead of (1)
  \[
  R^O = \{ A(r) \Rightarrow O \ l | r \in R^O[l, n] \} \cup \{ A(r) \Rightarrow O \ l | r \in R^O[\sim l, n] \} \quad (A.0.1)
  \]
- if $D + \Box I \sim l$, then instead of (3)
  \[
  R^{SI} = \{ A(r) \Rightarrow SI \ l | r \in R^{SI}[\sim l, n] \} \cup \{ A(r) \Rightarrow SI \ l | r \in R^{SI}[l, n] \} \quad (A.0.3)
  \]

Then $D \equiv D'$.

**Proof.** The demonstration follows the inductive base and inductive step of Lemma 35 where we consider the particular case $\Box = B$. Since here operations to obtain $D'$ modify only the consequent of rules, verifying conditions when a given rule is applicable/discarded reduces to clauses (2.1.2) and (4.1.2) of Definitions 7–8, while conditions for a rule being body-applicable/discarded are trivially treated. Moreover, the analysis is narrowed to modalities $O$, $I$, and $SI$ since rules for the other modalities are not affected by the operations of the lemma. Finally, notice that operations of the lemma do not erase rules from $R$ to $R'$ but it may be the case that, given a rule $r$, if removal or truncation operate on an element $c_k$ in $C(r)$, then $r \in R[l]$ while $r \notin R'[l]$ for a given literal $l$ (removal of $l$ or truncation at $c_k$).

$P(1) = +\Box_X q$, with $X \in \{O, I, SI\}$. We start by considering condition (2.2) of Definition 11 where a rule $r \in R_X[q, i]$ is applicable in $D$ at $i = 1$ and $P(1)$. In both cases when $q = l$ or $q \neq l$, $q$ is the first element of $C(r)$ since either we truncate chains at $l$, or we remove $\sim l$ from them. Therefore, $r$ is applicable in
$D'$ as well. In the other direction, if $r$ is applicable in $D'$ at 1 and $P(1)$, then $r \in R$ has either $q$ as the first element, or only $\neg l$ precedes $q$. In the first case $r$ is trivially applicable, while in the second case the applicability of $r$ follows from the hypothesis that $D \vdash +\partial l$ and $D \vdash +\partial_O \neg l$ if $r \in R^O$, or $D \vdash +\partial l$ and $D \vdash -\partial_O \neg l$ if $r \in R^S$.

Concerning condition (2.3.1) of Definition 11 there is no such rule $s$ in $R$, hence $s$ cannot be in $R'$ (we recall that at $P(1)$, $s$ cannot be discarded in $D$ because that would imply a previous step in the proof). Regarding the other direction, we have to consider the situation where there is a rule $s \in R^Y[\neg q,j]$ which is not in $R'^Y[\neg q]$. This is the case when the truncation has operated on $s \in R^Y[\neg q,j]$ since $l$ preceded $\neg q$ in $C(s)$, making $s$ discarded in $D$ as well (either when (i) $Y = O$ or $Y = I$, or (ii) $D \vdash -\partial_O \neg l$ and $Y = SI$).

For (2.3.2) the reasoning is the same of the equivalent case in Lemma 35 with the additional condition that rule $t$ may be applicable in $D'$ at $P(1)$ but $q$ appears at index 2 in $C(t)$ in $D$.

$P(n + 1) = +\partial_X q$, with $X \in \{O, I, SI\}$. Again, let us suppose $r \in R[q,i]$ to be applicable in $D$ for $X$ at $i$ and $P(n + 1)$. By hypothesis and clauses (2.1.2) or (4.1.2) of Definition 7, we conclude that $c_k \neq l$ and $q \neq \neg l$ (Conflict(B,I) and Conflict(B,SI)). Thus, $r$ is applicable in $D'$ by inductive hypothesis. The other direction sees $r \in R'[q,i]$ applicable in $D'$ and either $\neg l$ preceded $q$ in $C(r)$ in $D$, or not. Since in the first case, the corresponding operation of the lemma is the removal of $\neg l$ from $C(r)$, while in the latter case no operations on the consequent are done, the applicability of $r$ in $D$ at $P(n + 1)$ is straightforward.

For condition (2.3.1), the only difference between the inductive base is when there is a rule $s$ in $R^Y[\neg q,j]$ but $s \not\in R'^Y[\neg q,k]$. This means that $l$ precedes $\neg q$ in $C(s)$ in $D$, and thus by hypothesis $s$ is discarded in $D$. Notice that if $q = l$, then $R'^Y[\neg l,k] = \emptyset$ for any $k$ by the removal operation of the lemma, and thus condition (2.3.1) is vacuously true.

$P(1) = -\partial_X q$ and $P(n + 1) = -\partial_X q$, with $X \in MOD$. They trivially follow from the inductive base and inductive step. \hfill \Box

**Lemma 38.** Let $D = (F, R, >)$ be a modal defeasible theory such that $D \vdash -\partial l$ and $D' = (F, R', >)$ be the theory obtained from $D$ where

$$R'^l = \{ A(r) \Rightarrow C(r)! \neg l \mid r \in R^l[\neg l, n] \}.$$ 

Moreover,
• if $D \vdash +\partial_{\mathcal{O}} l$, then
  \[ R^\mathcal{O} = \{ A(r) \Rightarrow_{\mathcal{O}} C(r) \land l \mid r \in R^\mathcal{O}[l, n] \}; \]

• if $D \vdash -\partial_{\mathcal{O}} l$, then
  \[ R^\mathcal{SI} = \{ A(r) \Rightarrow_{\mathcal{SI}} C(r) \land \neg l \mid r \in R^\mathcal{SI}[\neg l, n] \}. \]

Then $D \equiv D'$.

Proof. The demonstration is a mere variant of that of Lemma 37 since: (i) Proposition 16 states that $+\partial_X m$ implies $-\partial_X \neg m$ (mode $D$ is not involved), and (ii) Operations of the lemma are a subset of those of Lemma 37 where we switch $l$ with $\neg l$, and the other way around. \hfill \Box

Lemma 39. Let $D = (F, R, \succ)$ be a modal defeasible theory such that $D \vdash +\partial_{\mathcal{O}} l$ and $D' = (F, R', \succ)$ be the theory obtained from $D$ where

\begin{align*}
R^\mathcal{O} &= \{ A(r) \Rightarrow_{\mathcal{O}} C(r) \land \neg l \land \neg l \mid r \in R^\mathcal{O}[\neg l, n] \} \quad \text{(A.0.1)} \\
R^\mathcal{SI} &= \{ A(r) \Rightarrow_{\mathcal{SI}} C(r) \land \neg l \mid r \in R^\mathcal{SI}[\neg l, n] \}. \quad \text{(A.0.2)}
\end{align*}

Moreover,

• if $D \vdash -\partial l$, then instead of (1)
  \[ R^\mathcal{O} = \{ A(r) \Rightarrow_{\mathcal{O}} C(r) \land \neg l \land \neg l \mid r \in R^\mathcal{O}[\neg l, n] \} \cup \{ A(r) \Rightarrow_{\mathcal{O}} C(r) \land l \mid r \in R^\mathcal{O}[l, n] \}; \quad \text{(A.0.1)} \]

• if $D \vdash -\partial \neg l$, then instead of (2)
  \[ R^\mathcal{SI} = \{ A(r) \Rightarrow_{\mathcal{SI}} C(r) \land \neg l \mid r \in R^\mathcal{SI}[\neg l, n] \} \cup \{ A(r) \Rightarrow_{\mathcal{SI}} C(r) \land l \mid r \in R^\mathcal{SI}[l, n] \}. \quad \text{(A.0.2)} \]

Then $D \equiv D'$.

Proof. Again, the proof is a variant of that of Lemma 37 that differs only when truncation and removal operate on a consequent at the same time.

A CTD is relevant whenever its elements are proved as obligations. Consequently, if $D$ proves $\mathcal{O}l$, then $\mathcal{O}\neg l$ cannot hold. If this is the case, then $\mathcal{O}\neg l$ cannot be violated and elements following $\neg l$ in obligation rules cannot be triggered. Nonetheless, the inductive base and inductive step do not significantly
differ from those of Lemma 37. In fact, even operation (1) involving truncation and removal of \( \sim l \) does not affect the equivalence of conditions for being applicable/discarded between \( D \) and \( D' \).

Proofs for Lemmas 40–44 are not reported. As stated for Lemma 39, they are variants of that for Lemma 37 where the modifications concern the set of rules on which we operate. The underlying motivation is that truncation and removal operations affect when a rule is applicable/discarded as shown before where we have proved that, given a rule \( s \) and a literal \( \sim q \), it may be the case that \( \sim q \notin C(s) \) in \( R' \) while the opposite holds in \( R \). Such modifications reflect only the nature of the operations of truncation and removal while they do not depend on the mode of the rule involved.

**Lemma 40.** Let \( D = (F, R, >) \) be a modal defeasible theory such that \( D \models \neg \partial_D l \) and \( D' = (F, R', >) \) be the theory obtained from \( D \) where

\[
R^D = \{ A(r) \Rightarrow_D C(r) \mid \sim l \models r \in R^D[l, n] \}.
\]

Moreover,

- if \( D \models \neg \partial l \), then

\[
R^D = \{ A(r) \Rightarrow_D C(r) \mid \sim l \models r \in R^D[l, n] \}.
\]

Then \( D \equiv D' \).

**Lemma 41.** Let \( D = (F, R, >) \) be a modal defeasible theory such that \( D \models +\partial_D l \), \( D \models +\partial_D \sim l \), and \( D' = (F, R', >) \) be the theory obtained from \( D \) where

\[
R^D = \{ A(r) \Rightarrow_G C(r) \mid \sim l \models r \in R^D[l, n] \}
\]

Then \( D \equiv D' \).

**Lemma 42.** Let \( D = (F, R, >) \) be a modal defeasible theory such that \( D \models \neg \partial_D l \) and \( D' = (F, R', >) \) be the theory obtained from \( D \) where

\[
R^D = \{ A(r) \Rightarrow_D C(r) \mid \sim l \models r \in R^D[l, n] \}
\]

Then \( D \equiv D' \).

**Lemma 43.** Let \( D = (F, R, >) \) be a modal defeasible theory such that \( D \models +\partial_X l \),
with \( X \in \{G, I, SI\} \), and \( D' = (F, R', >) \) be the theory obtained from \( D \) where

\[
R'^X = \{ A(r) \Rightarrow_X C(r) \mid r \in R^X[l, n] \} \cup \{ A(r) \Rightarrow_X C(r) \ominus l \mid r \in R^X[\neg l, n] \}.
\]

Then \( D \equiv D' \).

**Lemma 44.** Let \( D = (F, R, >) \) be a modal defeasible theory such that \( D \vdash \neg \partial_X l \), with \( X \in \{G, I, SI\} \), and \( D' = (F, R', >) \) be the theory obtained from \( D \) where

\[
R'^X = \{ A(r) \Rightarrow_X C(r) \ominus l \mid r \in R^X[l, n] \}.
\]

Then \( D \equiv D' \).

**Lemma 45.** Let \( D = (F, R, >) \) be a modal defeasible theory and \( l \in \text{Lit} \) such that (i) \( Xl \not\in F \), (ii) \( \neg Xl \not\in F \) and \( Yl \cap F = \emptyset \) with \( Y = X \) or Conflict\((Y, X)\), (iii) \( \exists r \in R^X[l, 1] \cup R^B_X[l, 1] \), (iv) \( A(r) = \emptyset \), and (v) \( R^X[\neg l] \cup R^B_X[\neg l] \cup R_Y[\neg l] \setminus R_{inf} \subseteq r_{inf} \), with \( X \in \text{MOD} \setminus \{D\} \). Then \( D \vdash +\partial_X l \).

**Proof.** To prove \( xl \), Definition 11 must be taken into consideration. Since hypothesis (i) falsifies clause (1), then clause (2) must be the case. Let \( r \) be a rule that meets the conditions of the lemma. Hypotheses (iii) and (iv) state that \( r \) is applicable for \( X \). In particular, if \( r = s^\circ \in R^B_X \) then \( s \) is Conv-applicable. Finally, for clause (2.3) we have that all rules for \( \neg l \) are inferiorly defeated by an appropriate rule with empty antecedent for \( l \), but a rule with empty body is applicable. Consequently, all clauses for proving \( +\partial_X \) are satisfied. Thus, \( D \vdash +\partial_X l \). □

**Lemma 46.** Let \( D = (F, R, >) \) be a modal defeasible theory and \( l \in \text{Lit} \) such that (i) \( DL \not\in F \), (ii) \( \neg DL \not\in F \), (iii) \( \exists r \in R^D[l, 1] \cup R^B^D[l, 1] \), (iv) \( A(r) = \emptyset \), and (v) \( r_{sup} = 0 \). Then \( D \vdash +\partial_D l \).

**Proof.** The demonstration is analogous to that for Lemma 45 since all lemma’s hypotheses meet clause (2) of Definition 9. □

**Lemma 47.** Let \( D = (F, R, >) \) be a modal defeasible theory and \( l \in \text{Lit} \) such that \( Xl \not\in F \) and \( R^X[l] \cup R^B_X[l] = \emptyset \), with \( X \in \text{MOD} \). Then \( D \vdash \neg \partial_X l \).

**Proof.** Conditions (1) and (2.2) of Definitions 10 and 12 are vacuously satisfied with the same comment for \( R^B_X \) in Lemma 45. □

**Lemma 48.** Let \( D = (F, R, >) \) be a modal defeasible theory and \( l \in \text{Lit} \) such that (i) \( X \lnot l \not\in F \), (ii) \( \neg X \lnot l \not\in F \) and \( Yl \not\in F \) with \( X = Y \) or Conflict\((Y, X)\), (iii)
\[ \exists r \in R^X[l,1] \cup R^{A,X}[l,1], (iv) A(r) = \emptyset, \text{ and (v) } r_{sup} = \emptyset, \text{ with } X \in \text{MOD}. \]

Then \( D \vdash -\partial_X \lnot l \).

Proof. Let \( r \) be a rule in a theory \( D \) for which the conditions of the lemma hold. It is easy to verify that clauses (1) and (2.3) of Definitions 10 and 12 are satisfied for \( \lnot l \). \( \blacksquare \)
RUNNING EXAMPLE OF ALGORITHM 6

We now detail how Algorithm 6 splitPattern works by reporting a toy theory, specifically the same proposed in Example 14, and showing each step of its execution. The initial point is the graph resulting from the execution of Algorithm 4 complianceByDesign up to Line 32. To ease the reading, we report below Example 14 along with Figure 6.5 and 6.6.

Example 15. Let $D_2 = (\{t_1, \ldots, t_7, u_1, \ldots, u_5\}, R, \emptyset)$ be a theory such that

$$R = \{r_1 : t_1, t_2 \Rightarrow u_1, r_2 : t_1, t_2, t_3 \Rightarrow u_2, r_3 : t_1, t_4, t_5 \Rightarrow u_3, r_4 : t_1, t_4, t_6 \Rightarrow u_4, r_5 : t_1, t_5, t_7 \Rightarrow u_5, r_6 : t_1, t_2, t_3, t_8 \Rightarrow u_6\}.$$

Alg. 4, Line 33: $Vlist = \{t_1, t_2, t_3, t_4, t_5\}$
Alg. 4, Line 34: $l = t_1$
Alg. 4, Line 35: $sL = \{(t_1, t_2), (t_1, t_2, t_3), (t_1, t_4, t_5), (t_1, t_4, t_6), (t_1, t_5, t_7), (t_1, t_2, t_3, t_8)\}$
Alg. 6, Line 2: $candGlobal = \{l\}$
Alg. 6, Line 3: $elem_{sL} = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8\}$
Alg. 6, for at Lines 4-7:

- $setsA_{t_1} = \{(t_1, t_2), (t_1, t_2, t_3), (t_1, t_4, t_5), (t_1, t_4, t_6), (t_1, t_5, t_7), (t_1, t_2, t_3, t_8)\}$
- $intersect_{t_1} = \{t_1\}$
- $setsA_{t_2} = \{(t_1, t_2), (t_1, t_2, t_3), (t_1, t_2, t_3, t_8)\}$
- $intersect_{t_2} = \{t_1, t_2\}$
- $setsA_{t_3} = \{(t_1, t_2, t_3), (t_1, t_2, t_3, t_8)\}$
- $intersect_{t_3} = \{t_1, t_2, t_3\}$
- $setsA_{t_4} = \{(t_1, t_4, t_5), (t_1, t_4, t_6)\}$
- $intersect_{t_4} = \{t_1, t_4\}$
APPENDIX B. RUNNING EXAMPLE OF ALGORITHM 6

- $sets_{A_{t_5}} = \{(t_1, t_4, t_5), \{t_1, t_5, t_7\}\}
- intersect_{t_5} = \{t_1, t_5\}
- $sets_{A_{t_6}} = \{(t_1, t_4, t_6)\}
- intersect_{t_6} = \{t_1, t_4, t_6\}
- $sets_{A_{t_7}} = \{(t_1, t_5, t_7)\}
- intersect_{t_7} = \{t_1, t_5, t_7\}
- $sets_{A_{t_8}} = \{(t_1, t_2, t_3, t_8)\}
- intersect_{t_8} = \{t_1, t_2, t_3, t_8\}

Alg. 6, Line 8: $candidates = \{t_2, t_3, t_4, t_5\}$

Alg. 6, for at Lines 12-45: iteration on $a = t_2$

- $a = t_2$, Line 13: $candGlobal = \{t_1, t_2\}$
- $a = t_2$, Line 14: $S' = \{t_1\}$
- $a = t_2$, Line 18: $S = \{t_1, t_2\}$
- $a = t_2$, Line 20: $supp_{sL} = \{(t_1, t_2), \{t_1, t_2, t_3\}, \{t_1, t_2, t_3, t_8\}\}$
- $a = t_2$, Line 21: $sL = \{(t_1, t_4, t_5), \{t_1, t_4, t_6\}, \{t_1, t_5, t_7\}\}$
- $a = t_2$, Line 30: $onlyS = \{t_1, t_2\}$
- $a = t_2$, after for at Lines 31-34: $candGlobal = \{t_1, t_2\}$
- $a = t_2$, Line 35: $sLrest = \{(t_1, t_2, t_3), \{t_1, t_2, t_3, t_8\}\}$
- $a = t_2$, Line 40: $sLrest = \{(t_3), \{t_3, t_8\}\}$
- $a = t_2$, Line 41: $m = t_3$
- $a = t_2$, Line 42: $splitPattern(t_3, \{(t_3), \{t_3, t_8\}\}, Or-ST_{t_1, t_2})$

Alg. 6, Line 2: $candGlobal = \{t_1, t_2, t_3\}$

Alg. 6, Line 3: $elem_{sL} = \{t_3, t_8\}$

Alg. 6, for at Lines 4-7:

- $sets_{A_{t_3}} = \{(t_3), \{t_3, t_8\}\}$
- intersect_{t_3} = \{t_3\}
- $sets_{A_{t_8}} = \{(t_3, t_8)\}$
- intersect_{t_8} = \{t_3, t_8\}

Alg. 6, Line 8: $candidates = \emptyset$

Alg. 6, Line 10: $candidates = \{t_3\}$

Alg. 6, for at Lines 12-45: iteration on $a = t_3$

- $a = t_3$, Line 13: $candGlobal = candGlobal$
- $a = t_3$, Line 14: $S' = \{t_3\}$
- $a = t_3$, Line 16: $S' = \{t_3\}$
- $a = t_3$, Line 20: $supp_{sL} = \{(t_3), \{t_3, t_8\}\}$
- $a = t_3$, Line 21: $sL = \emptyset$
\(a = t_3\), Line 30: only\(S = \{\{t_3\}, \{t_3, t_8\}\}\)
\(a = t_3\), after for at Lines 31-34: \(\text{candGlobal} = \{t_1, t_2, t_3, t_8\}\)
\(a = t_3\), Line 35: \(sL_{\text{rest}} = \emptyset\)
\(a = t_3\), Line 37: \(\text{candidates} = \emptyset\)
\(a = t_3\), Line 38: Break and go to Line 44

\(a = t_2\), Line 44: \(\text{candidates} = \{t_4, t_5\}\)

Alg. 6, Line 12: \(a = t_4\)
\(a = t_4\), Line 13: \(\text{candGlobal} = \{t_1, t_2, t_3, t_4, t_8\}\)
\(a = t_4\), Line 14: \(S' = \{t_1\}\)
\(a = t_4\), Line 18: \(S = \{t_1, t_4\}\)
\(a = t_4\), Line 20: \(\text{supp}_{S_L} = \{\{t_1, t_4, t_5\}, \{t_1, t_4, t_6\}\}\)
\(a = t_4\), Line 21: \(S_L = \{t_1, t_4, t_7\}\)
\(a = t_4\), Line 30: only\(S = \{\{t_1, t_4, t_5\}, \{t_1, t_4, t_6\}\}\)
\(a = t_4\), after for at Lines 31-34: \(\text{candGlobal} = \{t_1, t_2, t_3, t_4, t_5, t_6, t_8\}\)
\(a = t_4\), Line 35: \(sL_{\text{rest}} = \emptyset\)
\(a = t_4\), Line 37: \(\text{candidates} = \emptyset\)
\(a = t_4\), Line 38: Break and go to Line 44

Exit
Alg. 4, Line 38: \(Vlist = \emptyset\)

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{process_graph.png}
\caption{Process graph of theory \(D'\) resulting from Algorithm 4 \textsc{compliance-ByDesign}: Run of the algorithm up to Line 33}
\end{figure}

\textbf{Figure B.1.}: Process graph of theory \(D'\) resulting from Algorithm 4 \textsc{compliance-ByDesign}: Run of the algorithm up to Line 33
**Figure B.2.** Process graph of theory $D'$ resulting from Algorithm 4 COMPLIANCE-ByDESIGN: Final process graph


G. Governatori, J. Hulstijn, R. Riveret, and A. Rotolo. On the representation of deadlines in a rental agreement. In A. R. Lodder and L. Mommers, editors,


J. Grant, S. Kraus, and M. Wooldridge. Intentions in equilibrium. In Fox and Poole [2010].


J. Hoffmann, I. Weber, and F. M. Kraft. Sap speaks PDDL. In Fox and Poole [2010].


Queste pagine sono in realtà dedicate

A Mariana del alma mia.
Por Amor.

A Simone. Per tutto.

Ai miei figli. Presenti e che verranno.
Perché questo lavoro possa ispirarvi ad essere migliori di me.
(Se non si è capito, voglio un Nobel in famiglia)
Ricordate sempre: Per aspera ad astra.