knowledge

MDPI

Article

A Comparative Study of Compliance Management Frameworks: PENELOPE vs. PCL

Ho-Pun Lam [1,*,†] **and Mustafa Hashmi** [2,3,†]

[1] Independent Researcher, Sydney, NSW 2015, Australia
[2] La Trobe LawTech, La Trobe Law School, La Trobe University, Bundoora VIC 3086, Australia
[3] Institute of Law and Technology, Autonomous University of Barcelona, 08193 Bellaterra, Spain
* Correspondence: oleklam@gmail.com
† These authors contributed equally to this work.

Abstract: Due to pressure from regulatory authorities, the requirement to remain compliant has tremendously increased over the last decade. To support compliance-related activities, a plethora of compliance management frameworks (CMFs), compliance languages and systems have emerged, which is on one hand advantageous, but may cause confusion when deciding which CMF can be used to best fulfil the organisation's internal requirements. This is due to the lack of acceptable compliance tools and methodologies in the compliance domain to uncover and compare the multidimensionality of capability between different frameworks and users' needs, which give raise to the question of how to formally evaluate a CMF. In this paper, we propose methodologies to formally evaluate CMFs, compliance languages and systems, in particular the underlying formal language of a CMF; and present the formal evaluation of two prominent formal language-based CMFs, namely, *PENELOPE* and *PCL*, with a business contract using formal analysis approach. Our evaluations formally validate that the proposed methodologies are instrumental in deciding on the suitability of a CMF when is comes to evaluating the underlying formal logic of the framework to represent different types of norms.

Keywords: norms; norms modeling; formal evaluations; compliance frameworks

Citation: Lam, H.-P.; Hashmi, M. A Comparative Study of Compliance Management Frameworks: PENELOPE vs. PCL. *Knowledge* **2022**, 2, 618–651. https://doi.org/10.3390/knowledge2040036

Academic Editor: Constantin Bratianu

Received: 11 July 2022
Accepted: 25 October 2022
Published: 16 November 2022

1. Introduction

The requirement of being compliant is no longer an option. Over the past decades, the failure of large corporations, such as WorldCom, Enron, etc., had created a resounding impact on the world economy. As a result, different regulatory acts and standards, such Sarbanex-Oxley Act [1] and BASEL regulation [2], have emerged to alleviate the problems that may appear, leaving organisations no choice but to comply with these regulations or face severe penalties [3], which introduces new challenges at both organisational and technical levels.

At the core of any organisation are its business processes, which provide an abstract view of the state-of-affairs of how well their businesses are running and their alignment with the regulatory requirements that govern the business operations. There are, in general, three strategies that enterprises can use to ensure that their businesses are compliant with the requirements, namely: (i) *design-time* compliance verification, which adheres to the *Compliance by Design* (CbD) principle [4], evaluates business processes at the design phase for potential non-compliant behaviour before they are executed; (ii) *run-time monitoring*, which evaluate the business process and identify critical inefficiencies and bottlenecks during its execution; and (iii) *auditing* or *post-execution*, which detect anomalies and violations after execution (see [5] for details).

To support these strategies, various compliance management frameworks (CMFs) offering different kinds of functionalities and capabilities have been proposed [6]. As stated in [7], "*the generalizability of a theory to a description of the results that the practitioner would*

observe if he were to use the theory in a new setting [. . .] is arguably the most important form of generalizability". For a CMF to be effective, it should be based on some conceptually sound foundations to faithfully capture the meanings of different types of norms [3], or the effectiveness of the CMF will be reduced significantly. Hence, the effectiveness of a CMF is largely dependent on the *expressive power* of the underlying formalism being used—whether it is capable of representing *all* types of norms, and how *accurately* it can capture the meanings of the norms.

In this paper, we present a methodology to evaluate CMFs, with a specific focus on how well the underlying formalisms can be used to represent different types of norms—in particular, how accurate the extracted norms can truly capture the intended meaning of the relevant business processes. For this purpose, we report here a formal evaluation of two prominent formal language-based compliance frameworks, namely: *Process Entailment for Elicitation of Obligations and Permissions* (PENELOPE) [8] and *Process Compliance Language* (PCL) [9], using the methodologies proposed in this paper to address the above-mentioned issues. The contribution of this paper is twofold:

(i) Methodology to *extract the normative requirements*: We propose a structured methodology to extract normative requirements, especially *obligations*, from the regulatory texts based on the well-known IF. . . THEN structure. To be able to capture different aspects of a business process, we extend the structure with *control-flow*, *data* and *resources*, and *temporal element*, making it possible to abstract the information of the regulatory texts to determine which rules are relevant to the activities of a process and under which conditions, which provides a basis for deeper analysis of the extracted norms.

(ii) Methodology to *compare CMFs*: Based on the classification model presented in [10], we proposed a methodology to formally evaluate the underlying formalism of CMFs in representing and evaluating the norms. Using a synthetic but plausible business contract, we model the normative requirements using PENELOPE and PCL, and examine their capabilities and limitations in encoding different types of norms as well as compliance verification. Our results show that the proposed approach allows us to determine the suitability of a CMF for representing legal norms and automate the compliance checking process in an effective and efficient way.

This paper is organized as follows. Section 2 recalls the basic normative requirements as proposed in [10,11], followed by a discussion on the nature and some characteristics of a typical business contract. The major components of the normative requirements extraction methodology will be described in Section 4. Based on this, a comparative analysis of the two frameworks mentioned above and a discussion on the shortcomings of the underlying formalisms used in these frameworks will be presented in Sections 5 and 7, respectively. Finally, an account of the related work is presented in Section 6, before concluding the paper with some remarks and pointers to future work.

2. Revisiting Normative Requirements

Norms in legal settings can be very complex and can include issues with local and global aspects, making it very difficult to extract and formulate it in a concise and unambiguous way.

Typically, norms aim to control the behaviour of their subject and prescribe the situations under which they are applicable and generate normative effects when applied. Within the scope of business process compliance (BPC), *deontic effects* (a.k.a. *normative positions*) are of interest, and can be classified into four basic categories, namely: (i) *obligation*: a situation, an act, or a course of action to which a bearer is legally bound, and a violation may appear if it cannot be achieved or the associated action cannot be performed; (ii) *prohibition*: a situation, an act, or a course of action which a bearer should avoid, and if it is achieved results in a violation; (iii) *permission*: a situation when no obligation or prohibition on the contrary holds; and (iv) *duty*: a situation or action required by a bearer's position or function undertaken through the binding force of something that is morally or legally right.

In general, a norm is *fulfilled* once its applicability conditions has been satisfied, and will remain *in force* until it has been terminated (or breached) and removed from the context, i.e., the time that the norm ceases to hold and will have no further interactions with other norms.

However, in BPC, a breach of an obligation (a.k.a. violation) does not necessarily mean the immediate termination of the obligation as there are circumstances that it can be compensated. Hence, the status of an obligation and the status of the business process may be subject to further investigations before a final conclusion can be decided.

The concept of permission has been extensively investigated in literature where a number of studies provided key ideas of the concept [12–16], logical understanding and interpretation in various settings [17–20], distinctions between different classes [18,21], and criticism [22–24] on permission applied to specific contexts have been reported. Permission can be classified into two kinds, namely: weak permission and strong permission. (Note that Makinson and van der Torre [18] labels weak and strong permissions as *negative* (−) and *positive* (+) permissions, respectively.) *Weak permission* is a straightforward notion to describe that something (object x) is permissible by a normative code if and only if its opposite (object $\neg$x) is not forbidden by the same code [18]; or in other words, it is implied from the absence of prohibition. *Strong permission*, in contrast, is rather more subtle and complex. It describes an object x is only permissible if and only if object x is explicitly permitted by the normative code, and are further divided as (i) *static permission*, which refers to something (object x) is permitted in the presence of another thing (object y) that is obligatory, and when joining them together we derive what is *explicitly* permitted by the normative system, and (ii) *dynamic permission*, which determinate "the limits on what may be forbidden without violating the static permission" [18]. (A formal analysis and causal normative treatment of strong permissions can be found at [18] and [25], respectively.)

It should be noted that permissions are constraints that cannot be violated. They play no specific role in the compliance validation process; rather, they are used to derive other deontic effects and obligations to the contrary, which can be used to prevent incorrect assessment of whether a business process is compliant [26].

The notion of duty has been been extensively explored in literature. It is somewhat similar to an obligation, which under some (contextual) conditions expects bearer's actions. However, what separates duties from obligations is that duties arise from bearer's societal position, status or role; whereas obligations may arise or be created voluntarily, and they may be owed to a citizen who has rights [27,28]. Brandt [29] describe duty and obligation as jointly distinctive based on (i) the nature of their content, which depends on the code and practice of the bearer's group; (ii) the conduct of one's duty or obligation is not required to be fixed by the nature of the bearer's action; and (iii) the force of '*duty-obligation*' is generally coercive (either morally or by law).

Similar to obligations, the conditions of a duty can be breached, which may result in moral or legal (potentially adverse) consequences. Depending on the nature and severity of the breach, remedial actions can be taken to remedy the breach. From a philosophical perspective, Kant argues that a breach of a duty is impossible. He based his claims on two grounds that (i) human being is necessitated to act morally on the call for the duty [12], and (ii) on acknowledging and acting upon the moral principles deemed relevant to the case [13]. However, Mill [14] argue against the impossibility of conflict of duty on moral grounds suggesting that violations of (moral) duty can be directly punished. It is because in Mills's view [14], duty is characterised as "*ought*" which directly corresponds to an obligation, and a violation of an obligation is punishable in law. We subscribe to this view in our classification. As the philosophical discussion on the concept of duty is out of the scope of this paper. Interested readers see for detailed account of the concept of duty, moral and non-moral characteristics, usage [27–30], correspondence with rights and obligations [31–33], conflict and compulsion of duty [12–14].

Each norm can have their lifespan and will produce effects when they become applicable. Through studying their temporal validity, i.e., the time of when a norm becomes

fulfilled, *holds*, being *used* and *violated* by other norms, refs. [10,11] proposed a classification framework to categories different types of norms. Here, we extend this framework by including their relationship with other deontic effects, such as duty, as depicted in Figure 1.

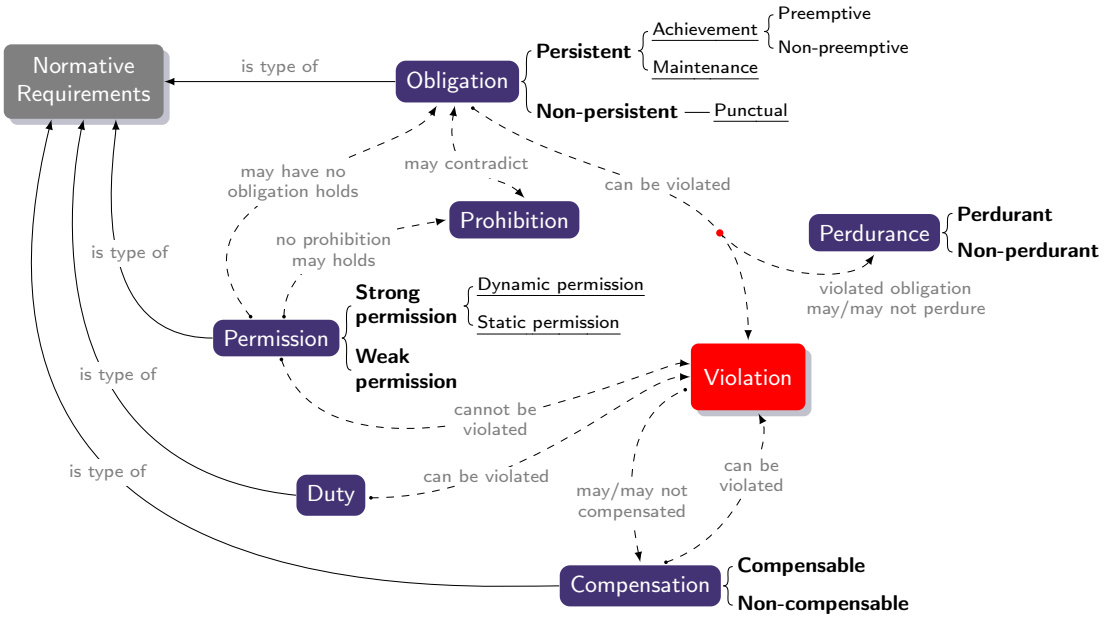

Figure 1. Classes and Relationships of Normative Requirements (adapted from [11]).

In general, *persistence obligations* are the most common type of obligation that appears in a business process. That is, when the conditions of an obligation has been fulfilled, it will remain in force until it has been terminated. It can further be classified into two types, namely: *achievement obligation*, meaning that it is sufficient if the contents of the obligation can be achieved at least once during the whole time interval; and a *maintenance obligation* requires the conditions of the obligation must be fulfilled during the whole time interval in which the obligation holds, irrespective to their type. If the obligation conditions of an achievement obligation are fulfilled even before the obligation starts to hold, then it will be classified as a *preemptive obligation*; otherwise, it is a *non-preemptive obligation*. A *punctual obligation*, on the other hand, requires the obligation to be fulfilled at a particular instance of time, or a violation will appear.

As mentioned before, one property of the obligations is that they can be violated. Depending on the violation conditions, a violated obligation may possibly be compensated by another obligation(s), keeping the business process remain compliant but operating under a sub-ideal situation. However, since the compensated obligations by themselves are obligations, they can further be violated and compensated recursively [34], which is semantically equivalent to the Contrary-to-Duty (CTD) structure [35]. That is, a situation where an obligation comes into effect while another (normally, more preferred) obligation is violated [36].

Mostly in literature, the CTD structure is considered as a kind of reparative approach to violation and is not uncommon in business contracts. In fact, many business contracts contain compensatory provisions, prescribing penalties and other sanctions that are enforced upon the breach of certain obligatory conditions.

However, it should be noted that not all obligations can be compensated, and uncompensated violations would result in the process being non-compliant or a penalty will be imposed. Therefore, in certain cases, it is a requirement that the effects of a violation must be considered during the compliance verification process. If an obligation persists after a violation, it is called a *perdurant obligation*; or a *non-perdurant obligation*, otherwise.

In the next section, we are going to describe the properties and characteristics of a business contract.

3. Characteristics and Properties of Legal Documents

Dealing with contracts or public regulations is part of the exercise that every organisation needs to articulate or implement in their own business processes. A *contract*, in general, is a specific agreement between two parties, namely: *offer* and *acceptance*, and must include (i) an *intention to create* a legal relation, (ii) a *consideration*—something of value (not necessarily monetary) that includes other to enter into the agreement, (iii) a *mutuality of obligation* where both parties must be bounded to perform their obligations, and the prescribed conditions constraining their behaviour (such as what they should do and what they should avoid, and it may also include their rights and duties while the contract is enforced), and (iv) the *competency and capacity* which states the legal capacity of the parties involved [37].

At a more abstract level, a typical business contract (or government regulation) may consists of the following document structure:

Part (Sections, Regulations, Clauses) – # or name
Subpart (subsection, sub-regulation, subclauses)
Paragraph (# or name)
Subparagraph (# or name)
Subsubparagraph (# or name)

In this sense, a contract can be considered as a finite set of articles (agreed and signed by the involved parties) under which each article (i.e., contract condition) can further be divided into different clauses as well as sub-clauses. Essentially, this structure allows the cross-referencing and backtracking of the norms, and indices of various components (e.g., sections, sub-sections, etc.) of the document [38].

Figure 2 presents a simplified version of a "*technology equipment supply*" business contract. As can be seen from the figure, the contract enters into force as soon as it is signed by the two parties, i.e., the *principal* (the contracting entity) and the *contractor* (the contracted entity); and will be terminated upon the fulfilment of agreed conditions. In addition, it also lays out the conditions under which both parties need to comply with.

Section 1: Contract *parties/persons* involved. . . .

Section 2: For the supply of the technology equipment, the Provider must deliver completely new and state-of-the-art equipment within 7 working days from the time when contract is signed. The Provider is responsible for all transportation and labour charges and must submit an Equipment Delivery Report (EDR) within 3 working days of the delivery. The Principal must provide an unimpeded access to the site where the equipment have to be delivered.

Section 3: (Provisions for Making Payment)
The Principal shall be responsible for paying in full within 15 days after the Provider has presented an invoice for any payment claims. If the payment is not made by the deadline, then the Provider can charge the Principal a penalty of 3% interest per calendar day in addition to the amount on invoice. The defaulted invoice must be paid within the next 7 days. Another 2.5% per day penalty is applicable, for any subsequent defaults of the invoice. The contract is automatically terminated after 3 consecutive defaults of any payments. The Principal has the right to delay any payments pursuant to an unresolved complaint in case of any conflict; however, the Principal must pay any outstanding amount within 3 days from when the resolution is agreed upon.

Section 4: (Request to Change the Scope of the Contract)
The Principal can request any changes in the scope of the contract at any time for which Principal must make a formal request. The changes request can be either major or minor changes from the provisions rendering new requirements or changes in the specifications of the tendered equipment or some minor changes in the specifications, change of location, number of required items, etc. The Provider shall act upon accordingly on the change request, and can accept or reject the request. If Provider accepts a major change request, she must seek approval for new changes from Principal, who shall be responsible to pay the costs for major changes, whereas for minor change no costs can be claimed. Accordingly, all the requested changes must be documented in the contract as per the conditions of the contract thereof.

Figure 2. Simplified example of a technology equipment supply contract (adopted from [39]).

However, a contract may contain vague and ambiguous legal jargons or terms (with sometimes unstructured or even badly structured texts) which is difficult to understand. Typically, an analyst (in the industry) will look for the important information by scanning the legal document and make sense of it based on the reporter's model of *5Ws* and *1H*, as illustrated in Figure 3.

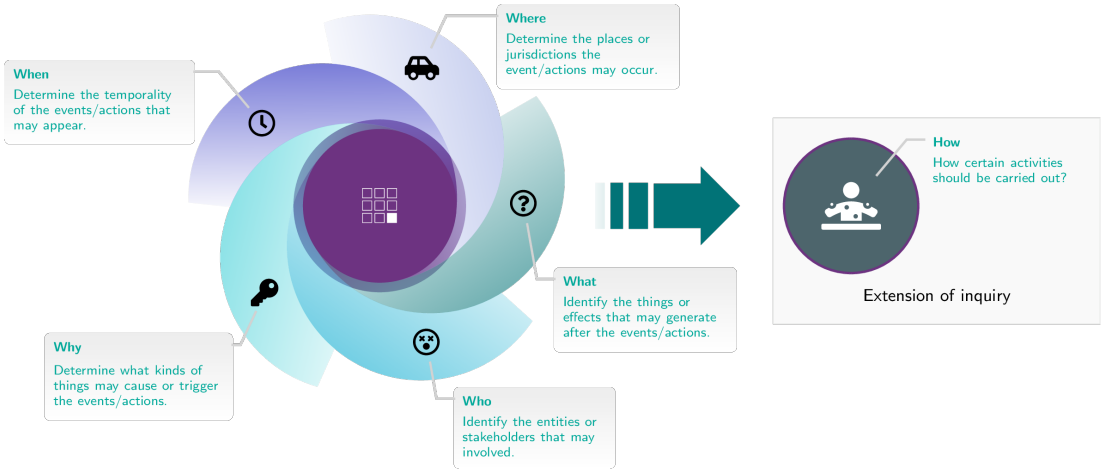

Figure 3. Reporter's model of inquiry.

Who: helps to identify who the document is from; who will be affected, or who will benefit. For example, people, objects, etc.

Why: may help to identify the goal, for example, causes for events or actions. Hence, an analyst should look for key terms or clues on 'what the norm aims to achieve'. For example, a subpoena clause might want an appearance in the court in case of a breach.

What: may facilitate the identification of actions of events. For example, a court notice may require a personal appearance in the court for a case. Hence, it's particularly important that the analyst pay special attention to what the norm asks for, i.e., is that anything that one needs to do?

When: may help to determine the temporality of some events or actions, meaning that a norm might prescribe some deadline of time limits in performing some action. For example, rent must be paid on the second (2nd) day of every month.

Where: may help to determine the places or jurisdictions a norm may be applicable at. For example, where the contract is applicable or which states have the jurisdiction. Accordingly, a norm might prescribe some context, which may also have some relationship with a course of actions. For example, where should the admission form be sent?

Whilst norms are prescriptive in nature, they may also define the course of actions on *how* a certain activity should be carried out. Hence, analysts should look for certain clues to determine *what* and/or *who* will be affected, or if there is a specific order (of tasks) that needs to be considered.

4. Norms Extraction Methodology

Based on the reporter's model presented in the previous section, in this section, we are going to present an approach to properly extract the legal norms from regulatory texts, which comprises of three distinct components, namely: (i) *document abstraction process*, (ii) *logical structure abstraction*, and (iii) *normative rules generation*, as illustrated in Figure 4.

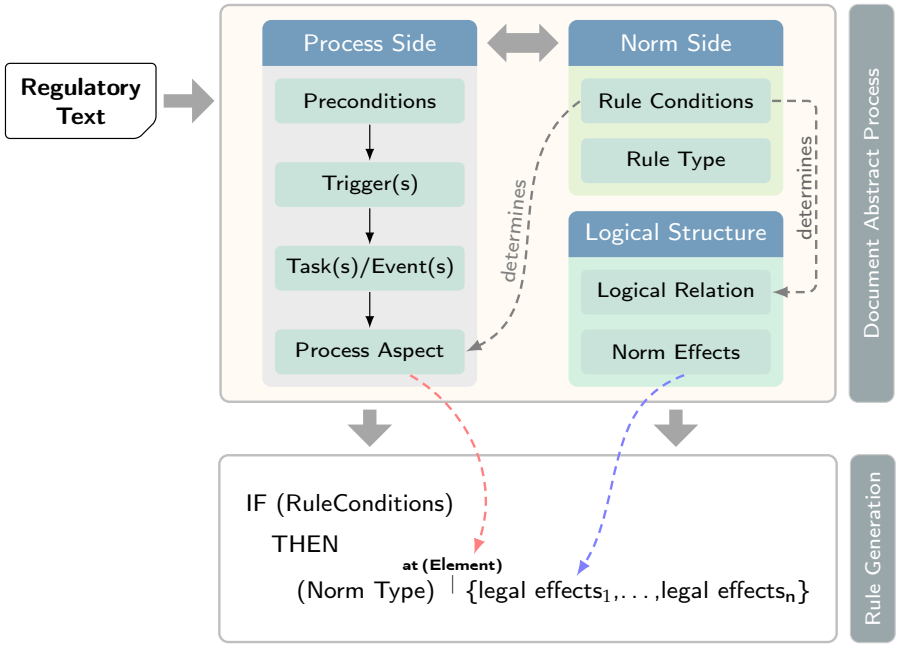

Figure 4. Norms extraction methodology architecture (adopted from [39]).

4.1. Document Abstraction Process

To improve the comprehensibility of a legal document for a better understanding of the contract conditions, we have to abstract and convert it into a structure that can facilitate the legal norms extraction process. Hence, we utilize the reporter's model presented in the previous section to abstract the logical structure of a legal document using the steps as shown below.

Identify preconditions: A *precondition* is a *predicate* that must always be true before an action can take place (or must appear before an operation in a formal specification). It provides the basis for the applicability of a particular rule with respect to a relevant context, and is a frame of reference that ensures the implementation of the rule. In some situations, it is paramount that a precondition must remain true all the time during the execution of an action, or the effects of such action will become undefined.

Identify triggers for rules: A *trigger* is the event(s) that can be used to trigger the initiation and/or termination of a rule. To determine the triggers, one should look for some temporal or substantive or semiotic indicators that must hold in the set of preconditions of the rule. Note that, in the legal domain, it is *not* uncommon that a rule can have one or more triggers associated with it [34].

Identify relevant tasks: A *task* is an activity that is carried out to achieve a specific goal, and can be divided into a set of assignments with defined deadlines. Generally, a task has a start and an end point determining the temporal validity of the execution of the task. The completion of all assignments indicates the completion of a task.

From a regulatory compliance perspective, it is possible that a norm may impose constraints on one or more tasks of the process controlling its behaviour by stipulating *when* and *how* an activity should be carried out. Hence, the analyst needs to ask *what (or which)* are the tasks a rule is applicable to; *when* (and *until when*) the rule is applicable; *how* the activity should be performed, etc. The *when* question determines the temporal validity of the rule giving the start and end conditions; *why* the rule is applicable to the task? The why question helps in defining the context in which a rule is applicable, and may determine the effects of the task and the legal effects the rule intends to achieve. Moreover, it may also determine dependencies between the rules and tasks of the process. Lastly, the *how* question may determine the ways in which a task should be carried out.

Identify process aspect: A typical business process is a self-contained temporal and logical model of a set of activities executed to achieve some defined goals. Essentially, a business process defines *when* a task should be carried out (control-flow), by *whom* (who is involved), and on *what* it needs to be done (to data and resources).
From a business process perspective, norms aim to control the behaviour of a business process by imposing constraints on one or more aspects of a business process. For example, a norm may prescribe conditions that an activity should be carried out in a particular order (e.g., acknowledge the complaint after receiving it); while others may prescribe some activities that need to be performed under a regular interval (e.g., monitoring the patient's vital signs during surgical operation). Hence, in this step, one needs to determine *which* process aspect a rule is applicable to, and whether the rule is relevant to one or more particular aspects.

Identify rule conditions: In addition to the items above, through performing or abstaining from some activities, a norm may stipulate one or more conditions on the business process that lead it to a desired state. For instance, some rules may prescribe actions that need to be repeated by a number of times (within a particular time frame); while others may prescribe some actions that need to be taken or avoided. Such conditions determine the objectives of the rules and provide details of intra-dependencies of tasks within a business process.

Identify rule type: According to its nature, norms in legal domain can be broadly classified into two main types, namely: (i) *regulative* or *prescriptive* norms, which describe the deontic behaviour of the contract, and (ii) *constitutive* norms, which regulate the creation of facts and modify the normative behaviour of the contract [15,40]. As different types of norms are generated in accordance with the rule types, identifying the rule types, thus, becomes a vital task in the rule generation process. Identifying the rules type wrongly not only affects the course of actions that are required by the rule to fulfil, but also imposes detrimental effects on the final outcomes of the whole system.

4.2. Logical Structure Abstraction

Abstracting the logical structure of a legal document aims to identify the interactions, and the intra- and inter-dependencies between different clauses that appear in the contract. In this sub-section, we are going to describe the logical structure abstraction process, which amounts to the following activities.

Determine Logical Relation: Tasks in a business process are normally executed in a coordinated manner and may possibly depend on each others during the execution, which is also true in a business contract. For instance, Clause 5.2 in YAWL Deed of Assignment (Warranties & Indemnity) (http://www.yawlfoundation.org/files/YAWLDeedOfAssignmentTemplate.pdf accessed on 20 June 2018) prescribes:

> *5.2 Each Contributor indemnifies and will defend the Foundation against any .....given by the Contributor under* **clause 5.1**.

which states the logical relation between Clauses 5.1 and 5.2. In some situations, a rule may also be used to specify that under certain circumstances a particular condition of a provision must be achieved; or the execution of that rule (and also the rules that follow it) should be suspended.
Essentially, such logical relations define the applicability (or absence) of a rule that might depends *fully*, or in some cases, *partially*, on the provision of the other rules, and can appear at provision level or between the propositions of the rule. In this sense, for a better understanding among the interactions of the rules that are being generated, it is crucial to identify the logical relations between different clauses within a contract.

Determine the Normative Effects: Norms intend to achieve a desired state or consequences. Essentially, the intension of the rules determines the type of normative effects that they want to achieve. As mentioned in the last subsection, depending on the applicability conditions and intension, norms in the legal domain can be classified into constitutive

norms and prescriptive norms, and can be generated from different types of rules accordingly. Depending on the applicability conditions and intension, the effects of the rule can be of different types. For example, a rule may ascribe the legal quality of person or object (qualificatory effects), some may impose obligation or impose restriction to refrain from a certain action (deontic effects), while other may state the modifications of norms such as substitution, abrogation or repeal (see [41] for other types of normative effects). Hence, in this step, the analysts should look for the desired effects that a rule is intended to achieve.

4.3. Normative Rules Generation: The IF... THEN Structure

Rules prescribed in statutory law can be very complex. In most cases, it may also prescribe certain constraints on actions in order to achieve some specific behaviour. In AI and legal domain, it is widely acknowledged that rules can be analysed according to their behaviour and the contexts under which they are created. Thus, rules, in general, have conditions-like structure [41,42], as shown below:

$$\mathsf{IF} \quad (\alpha_1, \ldots, \alpha_n) \quad \mathsf{THEN} \quad (\beta_1, \ldots, \beta_n) \tag{1}$$

where $\alpha_1, \ldots, \alpha_n$ and $\beta_1, \ldots, \beta_n$ are the antecedents (conditions) and the desired consequences (legal effects) of the rule, respectively. Note that, the antecedent of the norms can be empty meaning that their effects do not dependent on any condition. However, such norms may be deceptively ambiguous and limit the case of conditional norms in some situations [43].

In the context of BPC, norms may prescribe conditions relevant to different aspects of a business process. For example, a norm may intend to achieve some legal effects during a certain period of time (temporal) by involving some content (data), achieved by some agents (resources), and by following a certain order of operations (control-flow). Thus, in order to cater to the needs of different requirements that may appear in a business contract, we extend the IF... THEN structure by incorporating the control-flow, data, resources, and temporal elements into it, as illustrated in Equation (2).

$$\begin{aligned} &\mathsf{IF}\ (\mathsf{RuleConditions}) \\ &\quad \mathsf{THEN} \\ &\quad (\mathsf{NormType})^{\mid\ \text{at (Element)}}\ \{\mathsf{LegalEffects}_1, \ldots, \mathsf{LegalEffects}_n\} \end{aligned} \tag{2}$$

where $\mid^{\text{at (Element)}}$ is the element (which can be a task, an entity, data, resources, etc.) that the rule is applicable to, and $\mathsf{LegalEffects}_1, \ldots, \mathsf{LegalEffects}_n$ are the set of legal effects to be generated by the rule. Note that a rule might include one or more elements relevant to the tasks of the process.

4.4. Business Contract Norms Extraction

Based on the methodologies that we proposed above, in this section, we are going to extract the normative requirements from the business contract by examining its *preconditions* and *logical conditions*, which define the basis of interactions between obligations and control the behaviour of tasks involved in the contract.

We use the extended IF... THEN structure proposed in Equation (2) as a template to give a pseudo-representation of the rules being extracted from the contract, and generate the applicable conditions and their legal effects according to the entities and activities involved using the methodologies proposed in the sections before. For this purpose, our discussions will be based on §3 and §4 of the contract (cf. Section 3), which cover various type of deontic effects and contain clauses that are interacting with each others. (Note that, for the sake of clarity, we use §X to refer to the Section X that appear in the sample contract, and Section Y refers to the Section Y that appear in this paper.)

Intuitively, §3 of the contract gives us the following information.

Section 3 (Provisions for Making Payment)

(a) The Provider shall issue an invoice for making payment claims against the services provided, and
(b) The Principal is obliged to pay (in full) all the payments (and/or penalties) to Provider after receiving the invoice for services performed.
(c) Any payment(s) must be paid within 15 days from the date of receipt of the Provider's invoice (by the due date).
(d) Pursuant to Clause (3(c)), if the Principal fails to pay the amount within the due date of Provider's invoice, then those charges will be accrued late interest at the rate of 3% of the outstanding balance per day if the payment is made within 7 days of from the due date.
(e) If the payment is not made within 7 days from the due date of Provider's invoice, another 2.5% late interest per day on top of the rate imposed in Clause (3(d)) shall be imposed on the compounded amount of the invoice, and the payment must be made within next 10 working days.
(f) The contract is terminated for 3 defaults.
(g) Pursuant to Section 4 of the contract, the Principal may suspend or delay any payments until a resolution of any conflict(s) is agreed upon. However, the Principal shall make any suspended/outstanding payments within 3 working days from the day a resolution is agreed upon.

Hence, using the methodology proposed in Section 4.1 to abstract the document, we have:

Precondition: Services provided.
Trigger: An invoice was issued.
ProcessFragment: Figure 5 depicts the Business Process Modeling Notation (BPMN) model of the fragment of elicited activities for the terms of payment process, which is designed based on the interactions prescribed in the Section.

The following clausal conditions can be extracted from the clauses above:

C1. Provider must issue an invoice to claim any payments;
C2. Principal receives the issued invoice;
C3. Principal must pay received invoice (in full) within 15 days;
C4. If invoice deadline is violated, a 3% per day interest compensates it that must be paid within 7 days;
C5. If the defaulted invoice is violated another 2.5% interest compensating the violation must be paid within 10 days;
C6. The contract is terminated if 3 defaults of invoice occur;
C7. Subject to any unresolved conflict, the Principal can delay any payment to the Provider under clause (3(g));
C8. Suspended payments must be paid within 3 days if the resolution of the conflict is agreed upon.

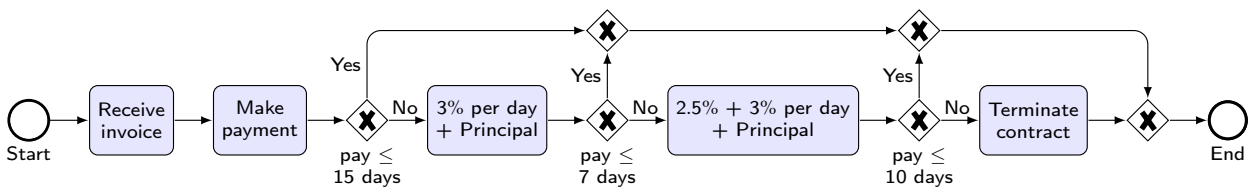

Figure 5. Terms of payment process fragment.

And details of the terms of payment-related activities and constraint types along with the process perspectives and responsible agents are illustrated in Table 1.

Based on the information above, next, we extract the logical conditions together with their effects for the applicable obligations prescribed in the contract.

$$\begin{array}{l}\text{IF } (\text{ProvideServices}, \text{PaymentClaims}) \\ \quad \text{THEN} \\ (\text{OAPNP}) \overset{\text{at (Provider)}}{\mid} \{\text{IssueInvoice}\}\end{array} \tag{LC1}$$

$$\begin{array}{l}\text{IF } (\text{PaymentClaims}, \text{IssueInvoice}) \\ \quad \text{THEN} \\ (-) \overset{\text{at (Principal)}}{\mid} \{\text{ReceivedInvoice}\}\end{array} \tag{LC2}$$

$$\begin{array}{l}\text{IF } (\text{PaymentClaims}, \text{IssueInvoice}, \text{ReceiveInvoice}) \\ \quad \text{THEN} \\ (\text{OAPNP}) \overset{\text{at (Principal, 15 Days)}}{\mid} \{\text{PayInvoice}\}\end{array} \tag{LC3}$$

Clauses (C1)–(C3) prescribe the conditions for the interactions and define logical dependencies between different clauses. In addition, they define which agents are involved and guide through ways how the activities should be carried out and the normative effects being produced during or after the interactions. For example, to make a payment claim, the Provider has to issue an invoice as illustrated in rule (LC1). On the other hand, on receiving the invoice, the Principal has a *non-preemptive*, *Persistent achievement obligation* (OAPNP) to pay the invoice within 15 days (rule (LC3)). The symbol "−" in rule (LC2) represents an arbitrary event, which can be anything, such as an obligation, an activity, a null value, etc.

Table 1. Activities and constraints for terms of payment.

Clause	Activity	Conditions	Agent	Process Aspect	Rule Type
C1	IssueInvoice	PaymentClaims, ProvideServices	Contractor	Resource	[OAPNP]
C2	ReceiveInvoice	IssuedInvoice	Principal	Resource	–
C3	PayInvoice	ReceivedInvoice	Principal	Resource, Time	[OAPNP]
C4	PayInvoice, Pay3%Interest	InvoiceDefault (Violation)	Principal	Resource, Data, Time	[OAPNP]
C5	PayCompoundInvoic Pay2.5%Interest	CompoundInvoicE (Violation)	Principal	Resource, Data, Time	[OAPNP]
C6	TerminateContract	3Defaults	Contractor	Resource	[OAPNP]
C7	DelayPayment	UnresolvedConflict	Principal	Resource	[Per]
C8	PayDelayedPayment	ConflictResolved	Principal	Resource, Time	[OAPNP]

To compensate a violation, first we have to have a violated obligation. Hence, we have to analyse the clauses and understand where the obligation has been violated, and how it can be compensated. So, based on Equation (2) below is the pseudo-representation of how a violated obligation and its compensation be represented.

$$\begin{array}{l}\text{IF } (\text{RulesConditions}) \\ \quad \text{THEN} \\ (\text{ViolatedObligation NormType}) \overset{\text{at (Element)}}{\mid} \{\text{PrimaryObligation}\} \\ \wedge\, (\text{CompensatedBy NormType}) \overset{\text{at (Element)}}{\mid} \{\text{SecondaryObligation}\}\end{array}$$

The meaning of the above rule is that, given the rule conditions, if the primary obligation is violated, that is, the effect $\neg PrimaryObligation^{\text{at (Element)}} \mid$ is generated (instead of $PrimaryObligation^{\text{at (Element)}} \mid$), then a secondary obligation will be triggered to compensate the violated obligation. Note, here it is possible that the newly enforced obligation may compensate the violated obligation by including *previous* as well as *new* conditions from the violated obligation, or it will *override* the violated obligation with giving some *totally new* conditions. Consequently, the clauses (C4)–(C6) can be modeled as follows.

$$\begin{aligned}&\text{IF } (\text{PaymentClaims}, \text{IssueInvoice}, \text{ReceiveInvoice}, \neg\text{PayInvoice})\\ &\quad \text{THEN}\\ &(\text{ViolatedObligation OAPNP})^{\text{at (Principal, 15 Days)}} \mid \{\text{PayInvoice}\}\\ &\wedge (\text{CompensatedBy OAPNP})^{\text{at (Principal, 7 Days)}} \mid \{\text{PayCompoundInvoice}\}\end{aligned} \tag{LC4}$$

$$\begin{aligned}&\text{IF } (\text{PaymentClaims}, \text{IssueInvoice}, \text{ReceiveInvoice}, \neg\text{PayInvoice}, \neg\text{PayCompoundInvoice})\\ &\quad \text{THEN}\\ &(\text{ViolatedObligation OAPNP})^{\text{at (Principal, 7 Days)}} \mid \{\text{PayCompoundInvoice}\}\\ &\wedge (\text{CompensatedBy OAPNP})^{\text{at (Principal,10 Days)}} \mid \{\text{PayCompoundInvoice\&Interest}\}\end{aligned} \tag{LC5}$$

$$\begin{aligned}&\text{IF } (\text{PaymentClaims}, \text{IssueInvoice}, \text{ReceiveInvoice},\\ &\qquad \neg\text{PayInvoice}, \neg\text{PayCompoundInvoice}, \neg\text{PayCompoundInvoiceWithInterest})\\ &\quad \text{THEN}\\ &(\text{ViolatedObligation OAPNP})^{\text{at (Principal, 10 Days)}} \mid \text{PayCompoundInvoice\&Interest}\\ &\wedge (\text{ViolationTerminable OAPNP})^{\text{at (Provider, 3 defaults)}} \mid \text{TerminateContract}\end{aligned} \tag{LC6}$$

Note here that, since the compensated obligations are themselves obligations (cf. Section 2), it can further be violated and compensated again in a recursive manner. This type of behaviour is captured by rules (LC4)–(LC6) such that the violated obligation in (LC4) is compensated by the obligation in (LC5); while the violated obligation in (LC5) is then further compensated by the obligation in (LC6). Besides, the logical conditions in (LC6) also capture the logical dependencies between these clauses and the data and resources being utilised.

As mentioned before, the interactions of different clauses within a contract can be very complex, across different sections, and a number of situations may occur, which can be important from the compliance checking perspective. For instance, under (C3), the Principal has the obligation to pay the invoice within 15 working days from the Provider's issue date. However, due to (C7), which is subject to clauses in §4 of the contract, the Principal has the permission to delay the payment if there is an unresolved conflict. Hence, we have the following rules.

$$\begin{aligned}&\text{IF } (\text{ReceiveInvoice}, \text{UnresolvedConflict})\\ &\quad \text{THEN}\\ &(\text{Per})^{\text{at (Principal, Time)}} \mid \{\text{DelayPayment}\}\end{aligned} \tag{LC7}$$

$$\begin{aligned}&\text{IF } (\text{ReceiveInvoice}, \text{UnresolvedConflict}, \text{DelayPayment}, \text{ResolvedConflict})\\ &\quad \text{THEN}\\ &(\text{OAPNP})^{\text{at (Principal, 3 Days)}} \mid \{\text{PayDelayPayment}\}\end{aligned} \tag{LC8}$$

(LC7) illustrates the permission to delay delaying payment in case of any unresolved conflict. In the structure, we do not have an exact value to the variable *time* as it refer to the time until when a resolution of the conflict is agreed upon as represented in (LC8), that the delayed payment must be made within 3 days after the resolution.

In a similar vein, §4 of the contract tells us the following information.

Section 4 (Request to Change the Scope of the Contract)

(a) The Principal can request any changes in the scope of the contract.
(b) Principal shall make a formal request to change the scope of the contract.
(c) Request changes can be either *major changes* or *minor changes*.
(d) Provider can accept or reject requested changes.
(e) If Provider accept major request, s/he shall seek approval from Principal.
(f) Principal shall pay for any accepted major changes to the contracted services.
(g) For any minor changes, Provider shall not claim any payment.
(h) All changes (major/minor) shall be documented.

Hence, by abstracting the document, we have:

Precondition : An inforce contract.
Trigger : Change in scope request.
ProcessFragment : Figure 6 depicts the process fragment of the elicited activities for the changes in scope request process. The following clausal conditions can be extracted from the above clauses:

C9. Principal can request changes in the scope of contract.
C10. Formal request must be made for any changes in the contract.
C11. A change request can be either:
(a) a *major change request*.
(b) a *minor change request*.
C12. Provider is permitted to accept or reject the request.
C13. For accepted major change request, Provider must seek Principal's approval.
C14. Principal must pay for accepted changes.
C15. Provider is prohibited to claim any costs for minor changes.
C16. All changes to the contract conditions must be documented.

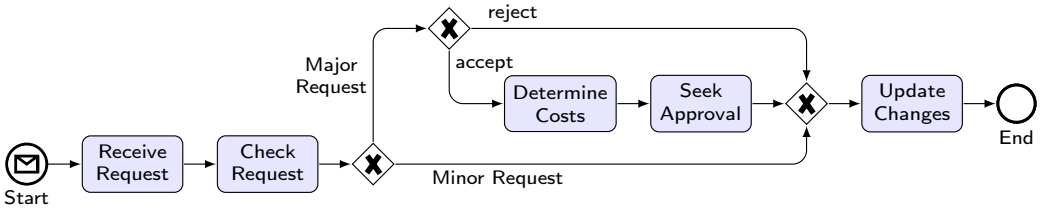

Figure 6. Changes in scope request fragment.

Table 2 summarizes the activities, responsible agents, and applicable constraints with their types for the request for changes process. Subsequently, the following can be generated, which include the logical conditions and legal effects of the applicable obligations for the above conditions.

$$\begin{array}{l}\mathsf{IF}\ (\mathsf{InForceContract})\\ \quad \mathsf{THEN}\\ \quad\ \ (\mathsf{Per})^{\ \mathbf{at}\ (\mathsf{Principal})} \mid \{\mathsf{RequestChange}\}\end{array} \tag{LC9}$$

$$\begin{array}{l}\mathsf{IF}\ (\mathsf{InForceContract})\\ \quad \mathsf{THEN}\\ \quad\ \ (\mathsf{OM})^{\ \mathbf{at}\ (\mathsf{Principal})} \mid \{\mathsf{MakeFormalChangeRequest}\}\end{array} \tag{LC10}$$

$$\text{IF } (\text{FormalChangeRequest}) \quad \text{THEN} \quad (\text{Per}) \overset{\text{at (Principal)}}{\mid} \{\text{MakeMajorChangeRequest} \vee \text{MakeMinorChangeRequest}\} \tag{LC11}$$

$$\text{IF } (\text{InForceContract}, \text{Received}(\text{Major} \vee \text{Minor})\text{Request}) \quad \text{THEN} \quad (\text{Per}) \overset{\text{at (Provider)}}{\mid} \{\text{AcceptRequest} \vee \text{RejectRequest}\} \tag{LC12}$$

$$\text{IF } (\text{InForceContract}, \text{ReceivedRequest}, \text{MajorRequest}) \quad \text{THEN} \quad (\text{OAPNP}) \overset{\text{at (Provider)}}{\mid} \{\text{SeekApproval}\} \tag{LC13}$$

$$\text{IF } (\text{InForceContract}, \text{MajorRequest}, \text{ApprovedChanges}) \quad \text{THEN} \quad (\text{OAPNP}) \overset{\text{at (Principal)}}{\mid} \{\text{PayCostsForAcceptedChanges}\} \tag{LC14}$$

$$\text{IF } (\text{InForceContract}, \text{ReceivedRequest}, \text{MinorRequest}) \quad \text{THEN} \quad (\text{OM}) \overset{\text{at (Provider)}}{\mid} \{\neg\text{ClaimCostForChanges}\} \tag{LC15}$$

$$\text{IF } (\text{InForceContract}, \text{ReceivedRequest}, (\text{Major} \vee \text{Minor}), \text{AcceptedRequest}) \quad \text{THEN} \quad (\text{OAPNP}) \overset{\text{at (Provider)}}{\mid} \{\text{DocumentChanges}\} \tag{LC16}$$

Table 2. Change request activities and constraints.

Clause	Activity	Conditions	Agent	Process Aspect	Rule Type
C9	ChangeRequest	Inforce Contract	Principal	Resource	[Per]
C10	ChangeRequest	Inforce Contract, MakeFormalRequest	Principal	Resource	[OM]
C11	ChangeRequest, FormalRequest	(MajorRequest ∨ MinorRequest)	Principal	Resource, CF	[Per]
C12	(Accept Request ∨ Reject Request)	FormalRequest, MajorRequest	Principal	Resource	[Per]
C13	SeekApproval	MajorRequest, AcceptedRequest	Provider	Resource	[OAPNP]
C14	PayforChanges	Accepted Changes, Approved Changes	Principal	Resource	[OAPNP]
C15	¬ClaimCosts	FormalChangeRequest, MinorRequest	Provider	Resource	[OM]
C16	UpdateDocument	AcceptedChanges	Principal, Provider	Resource	[OAPNP]

5. Evaluation of PENELOPE and PCL

In this section, we are going to discuss and evaluate the two prominent and widely used CMFs namely: PENELOPE and PCL. (Notice that, there are other CMFs reported in literature. For details we refer the reader to [3,5]) Our goal is to assess the feasibility and capabilities of different (logical) formalisms in modeling the normative requirements that are commonly found in business contracts.

In the following, we provide a short overview of the two frameworks and the underlying formalisms used. Then, based on their respective approaches, we will encode different types of norms that appear in the contract (cf. Section 3) and compare it using the methodologies presented in Section 4. For this purpose, we adopt the evaluation criteria described in [44].

5.1. Process Entailment for Elicitation of Obligations and Permissions (PENELOPE)

PENELOPE [8] is a declarative framework that declaratively captures obligations and permission constrains in the form of deontic assignments. It uses deontic properties for modelling obligations, permissions, and conditional commitments, as illustrated in Table 3, for compliance verification of business processes.

The framework uses a proprietary algorithm to generate the state space and control-flow of the business processes. The state space in the PENELOPE generated process corresponds to a set of permissions and obligations that are in force at a particular task. The interactions between the tasks flow in a linear fashion from one state to another. All states are enumerated until no obligation or permission is held at a state or if there is a violation which cannot be repaired.

Table 3. Deontic properties of PENELOPE (adopted from [8]).

Terms	Meanings
$\mathsf{Oblig}(ß, ff, \delta)$	agent π must do the activity α by due date δ
$\mathsf{Perm}(ß, ff, \delta)$	agent π can do the activity α prior to due date δ
$\mathsf{CC}(ß, ff_1, \delta_1, ff_2, \delta_2)$	agent π must do activity α_2 by due date δ_2 after activity α_1 is performed prior to the due date δ_1
(A) $\mathsf{Terminates}(ff, \mathsf{Oblig}(ß, ff, ffi), \tau) \longleftarrow \tau \leq \delta$	
(B) $\mathsf{Terminates}(ff, \mathsf{Perm}(ß, ff, ffi), \tau) \longleftarrow \tau \leq \delta$	
(C) $\mathsf{Happens}(\mathsf{Violation}(\mathsf{Oblig}(ß, ff, ffi)), \delta) \longleftarrow \mathsf{HoldsAt}(\mathsf{Oblig}(ß, ff, ffi), \delta) \wedge \sim\mathsf{Happens}(ff, \delta)$	
(D) $\mathsf{Initiates}(ff_1, \mathsf{Oblig}(ß_2, ff_2, ffi_2), \tau) \longleftarrow \tau \leq \delta_1 \wedge \mathsf{HoldsAt}(\mathsf{CC}(ß, ff_1, ffi_1, ff_2, ffi_2), \tau)$	

Once all state spaces are computed, the algorithm then draws the BPMN model for an agent of a business interaction. The tasks of the process are drawn whenever an obligation set contains all the obligations to fulfill by an agent in the activity. Since the modeling of business interactions between all participating agents is supported, any violations of obligations from a third party agent is drawn as intermediate time-out events; whereas the errors and end events are drawn if a violation of obligations or permissions by an agent (in a state) is found. With the designed compliant process models various types of inconsistencies, e.g., deontic, temporal and trust conflicts, can be identified. However, the generated process models are not meant to execute the process. It is rather to aid the process designers to check the impact of control-flow and timing constraints on business process design.

The deontic assignments in PENELOPE are modeled using Event Calculus (EC) [45], which provides a rich set of axioms for capturing the behaviour of dynamic occurrences of domain dependent and domain independent events. EC is based on the idea of states of time-varying properties of the world, called '*fluent*', which were initiated by some events at earlier time and hold at a particular time-point, and not terminated by some other events between that time-period. Accordingly, a fluent does not hold at some time if it was previously terminated and not resumed during that time [46].

The logic provides predicates expressing various types of states of event occurrences, e.g., Happens (occurrence of an event at a time-point), Initiates (an event triggers the property of a system),Terminates (an event terminates the property of the system), and HoldsAt (that the property of a system holds at a time-point). Also, some auxiliary predicates to express premature termination (Clipped) and resumption (Declipped) of an event at a particular point in time between the interval. The InitiallyTrue and InitiallyFalse allows the modeling

of the system's states where only partial information about the domain is available. In contrast, the domain independent axioms describe the states when a variable (fluent) holds or does not hold at a particular point in time.

For example, the axioms as in [47]:

$$\begin{aligned}\mathsf{HoldsAt}(\mathsf{P}, T_2) \longleftarrow\ & \mathsf{Happens}(\mathsf{P}, T_1) \wedge \mathsf{Initiates}(\mathsf{X}, \mathsf{P}, T_1) \\ & \wedge (T_1 < T_2) \wedge \neg\mathsf{Clipped}(T_1, \mathsf{P}, T_2)\end{aligned} \tag{E1}$$

The axiom (E1) states that the fluent P continues to hold until an event occurs that terminates it.

$$\begin{aligned}\neg\mathsf{HoldsAt}(\mathsf{P}, T_2) \longleftarrow\ & \mathsf{Happens}(\mathsf{X}, T_1) \wedge \mathsf{Terminates}(\mathsf{X}, \mathsf{P}, T_1) \\ & \wedge (T_1 < T_2) \wedge \neg\mathsf{Declipped}(T_1, \mathsf{P}, T_2)\end{aligned} \tag{E2}$$

Axiom (E2) states that the fluent P that has been terminated by the event X continues to hold until it is resumed (initiated) by some other event occurrence. The above axioms can be used to model the non-deterministic behaviour of the system. Hence, EC is used for modeling obligations that can be effected by unpredictable situations (see [47] for a detailed discussion on EC predicates).

Modeling Contractual Constraints with PENELOPE

Now we are ready to model the various obligation types of the business contract of Section 3.

Using the semantic properties of PENELOPE, we first model the clauses of making payments of the contract where the Provider issues the invoice to make any payment claims. Most of the clauses have dependency over other clauses, and thus, there exist a complex interactions between them. The logical conditions in rules (LC1) and (LC3).

To begin with, the logical conditions in rules (LC1) and (LC3) can be modeled as:

$$\begin{aligned}&\mathsf{Initiates}(\mathsf{ProvideService}, \mathsf{Oblig}(\mathsf{Provider}, \mathsf{IssueInvoice}, \mathsf{ffi}), \tau) \longleftarrow \\ &\qquad \mathsf{Happens}(\mathsf{ProvideService}, \tau) \wedge \mathsf{HoldsAt}(\mathsf{ProvideService}, \tau) \wedge \tau \leq \delta\end{aligned} \tag{E3}$$

$$\begin{aligned}&\mathsf{Initiates}(\mathsf{ReceiveInvoice}, \mathsf{Oblig}(\mathsf{Principal}, \mathsf{PayInvoice}, \mathsf{ffi}), \tau) \longleftarrow \\ &\qquad \mathsf{Happens}(\mathsf{IssueInvoice}, \tau) \wedge \mathsf{HoldsAt}(\mathsf{ProvideService}, \tau) \wedge \tau \leq \delta\end{aligned} \tag{E4}$$

From axiom (E3), we have $\mathsf{Initiates}(\mathsf{ProvideService}, \mathsf{Oblig}(\mathsf{Provider}, \mathsf{IssueInvoice}, \mathsf{ffi}), \tau)$, referring to an obligation that enters into force, when the fluent $(\mathit{ProvideService})$ holds at time τ, and is fulfilled by the due date δ. The event *ProvideService*, which happens at time τ triggers the obligation. Axiom (E4) states that given an issued invoice the principal has to pay the invoice at before the deadline δ, i.e., $\mathsf{Initiates}(\mathsf{ReceiveInvoice}, \mathsf{Oblig}(\mathsf{Principal}, \mathsf{PayInvoice}, \mathsf{ffi}), \tau)$, when the fluent provided service holds at τ, i.e., $\mathsf{HoldsAt}(\mathsf{ProvideService}, \tau)$.

There are a number of issues with the above representations. Firstly, the base predicate Initiates, by EC definition, does not initiate the deontic assignment (i.e., a norm) from the time when the triggering event occurs but from the next instant. Meaning that the effects of the deontic assignment do not enter into force at the time of event occurrence. In real situations, there can be cases when the effects of an obligation enter into force with the event occurrence. For example, the obligation to issue the invoice in clause (C1) immediately enters into force when the contract is signed, not from the next instant after signing it.

Secondly, there can be situations where the deontic assignment might not immediately enter into force when the event triggering the obligation occurs. This is due to the fact that the triggering of an event does not necessarily mean the *actual initiation* of a (legal) norm. In fact, the obligation (or in EC terms, the fluent) may not necessarily appear as an argument of the Initiates predicate.

Thirdly, in many cases, each obligation can have its own specific triggering events, and the occurrence of any of those triggering events can initiate the obligation. Besides, there could be a delay between the time when the triggering event occurs and the time when the

deontic assignment enters into force. Meaning that an obligation can be held deontically only when it is deontically initiated. However, with the current EC Initiates predicate, it is not possible to deontically trigger an obligation.

The obligation termination axiom in PENELOPE (cf. Table 3) is defined as follows:

$$\mathsf{Terminates}(\mathsf{ff}, \mathsf{Oblig}(\text{ß}, \mathsf{ff}, \mathsf{ffi}), \tau) \longleftarrow \tau \leq \delta \tag{E5}$$

which states that the obligation fluent, α, is terminated at time τ.

Now consider the logical conditions in rule (LC3) where the Principal has the obligation to pay the received invoice if she has received services under the contract. Hence, the terminating conditions for rule (LC3) can be modeled as below:

$$\begin{aligned}\mathsf{Terminates}(\mathsf{PaidInvoice}, \mathsf{Oblig}(\mathsf{Principal}, \mathsf{payinvoice}, \mathsf{ffi}), \tau) \longleftarrow \\ \mathsf{HoldsAt}(\mathsf{PaidInvoice}, \tau) \wedge \tau \leq \delta\end{aligned} \tag{E6}$$

meaning that the event *PaidInvoice* terminates the obligation fluent *PayInvoice* at τ.

Similar to the Initiates predicate, the issue with the Terminates predicate is that the obligation might possibly not terminate even during the time when the applicable conditions have been fulfilled. Here, there could be two reasons for that. The obligation is still relevant to other obligations that are in force in other tasks, i.e., having a dependency with other obligations; or the obligation has its own terminating events and triggering of one of those events is required for the termination. Hence, from the process compliance perspective, an obligation can be deontically terminated only if it is deontically initiated, and after the termination the obligation should cease its effects.

The issue here is that when using the Initiates predicate in PENELOPE, it would lead to temporal conflict such that both predicates, Initiates and Terminates, can be fired at the same time instant. For instance, from axiom (E4) we have $\mathsf{Initiates}(\mathit{ReceiveInvoice}, \mathsf{Oblig}(\mathsf{Principal}, \mathsf{PayInvoice}, \delta), \tau)$, where the event initiates the obligation fluent at time τ that must be fulfilled by the deadline δ; while the event *PayInvoice* (in the argument of the Terminates predicate of axiom (E6)) can also be fired at the same time instance τ, which leads to the immediate termination of the obligation fluent, and subsequently making the compliant verification process pointless.

This problem appears owing to the last condition of the Terminates predicates, regarding when an event can be terminated, i.e., $\tau \leq \delta$. A solution to this has been proposed by extending EC with deontic operators [34]. However, it seems that more work needs to be done before the PENELOPE community can adapt to it.

So far, we have discussed how obligations and permission can be represented in PENELOPE and some issues that may appear. In the following, we are going to examine how violations, and respectively, the CTD obligations, are handled in PENELOPE.

The logical conditions in rule (LC16) give the prohibition conditions forbidding the Provider to claim any payments from the Principal for any minor request. However, it is impossible to model such conditions directly in PENELOPE as it is operated under close world assumption [48], prohibitions are implicitly assumed when no obligation or permission can be derived.

However, one possible way to model prohibitions with PENELOPE semantics could be by modeling them as negative permissions. This is due to the fact, in most variants of deontic logic, *permissions* are considered as the lack of *obligations* and can be modeled as dual of each other, i.e., $\mathsf{O}\Delta \equiv \neg\mathsf{P}\neg\Delta$ and $\neg\mathsf{O}\neg\Delta \equiv \mathsf{P}\Delta$ [18,49,50]. This duality relation can be written as $\mathsf{O}\Delta \longrightarrow \mathsf{P}\Delta$ meaning that, if we have the obligation to do Δ, we also have the permission to do Δ [42] (note that the reverse may not be true). That is, if we are prohibited (F) to do Δ, then, logically speaking, we do not have the permission to do Δ, i.e., $\mathsf{F}\Delta = \neg\mathsf{P}\Delta$, or written formally, we have $\mathsf{F}\Delta = \mathsf{O}\neg\Delta$.

Hence, the prohibition conditions in rule (LC16) can be modeled as the negation of the permission of the original fluent, as shown below.

$$\begin{aligned}\mathsf{Initiates}(\mathsf{ChangeRequest}, \mathsf{Perm}(\mathsf{Contractor}, \neg\mathsf{ClaimCosts}, \mathsf{ffi}), \tau) \longleftarrow \\ \mathsf{Happens}(\mathsf{ChangeRequest}, \tau) \wedge \mathsf{HoldsAt}(\mathsf{MinorChangeRequest}, \tau') \wedge \tau \leq \delta\end{aligned} \tag{E7}$$

The meaning of axiom (E7) is that, given a minor change request, the prohibition, i.e., ¬PERM enters into force at state τ forbidding the Provider from claiming the costs for a minor change request. As now the logical conditions drawing a prohibition in rule (LC16) can be modeled, we may still not have the effects of the norm due to the above discussed problems with the Initiates and Terminates predicates thus, checking the compliance of above conditions may still not be possible.

The logical conditions in rule (LC8) give the conditions for both the cases of minor and major change request where Principal has the obligation to pay for the requested change can be collectively modeled as:

$$\begin{aligned}\mathsf{Initiates}(\mathsf{ChangeRequest}, \mathsf{Oblig}(\mathsf{Principal}, \mathsf{PayCostsForChanges}, \mathsf{ffi}), \tau) \longleftarrow \\ \mathsf{Happens}(\mathsf{ChangeRequest}, \tau) \wedge \mathsf{HoldsAt}((\mathsf{MajorRequest} \wedge \mathsf{ApprovedChanges}), \tau) \\ \wedge \tau \leq \delta\end{aligned} \tag{E8}$$

Axiom (E8) states that the obligation to pay costs for major changes is triggered when the major changes request is approved $\mathsf{HoldsAt}((MajorRequest \wedge ApprovedChanges), \tau)$ at time instance τ.

Next, we examine the case of violations handling, and the obligations arising from the violations of primary obligations. Reporting and handling violations of rules is one of the major requirements for a compliance management framework [51]. Timely reporting the violations allows the analysts to address the problems at the very beginning of the process design thus, saving lots of effort and time. Now, we look at how PENELOPE handles various violation situations, whether the notion of violations can be effectively handled with PENELOPE's violation semantics, and whether modeling the compensation (contrary-to-duty) obligation is possible.

The logical conditions in rules (LC4)–(LC6), which give the violation conditions and compensating (contrary-to-duty) obligations arising from the violations.

Consider the rule (LC4) where the Principal has the obligation to pay an invoice within 15 days, can be modeled as:

$$\begin{aligned}\mathsf{Initiates}(\mathsf{ReceiveInvoice}, \mathsf{Oblig}(\mathsf{Principal}, \mathsf{PayInvoice}, \mathsf{ffi}), \tau) \longleftarrow \\ \mathsf{Happens}(\mathsf{IssueInvoice}, \tau) \wedge \mathsf{HoldsAt}(\mathsf{ReceivedInvoice}, \tau) \wedge \tau \leq \delta\end{aligned} \tag{E9}$$

Axiom (E9) gives full instantiation of the interaction of the event, from where we get $\mathsf{Initiates}(ReceiveInvoice, \mathsf{Oblig}(\mathsf{Principal}, \mathsf{PayInvoice}, \delta), \tau)$ meaning that an obligation to pay the invoice enters into force at time τ that must be paid by the deadline δ.

Now, assume that the obligation to pay invoice is violated. Then, using the violation semantics mentioned in Table 3, we have:

$$\begin{aligned}\mathsf{Happens}(\mathsf{Violation}(\mathsf{Oblig}(ß, \mathsf{ff}, \mathsf{ffi})), \delta) \longleftarrow \\ \mathsf{HoldsAt}(\mathsf{Oblig}(ß, \mathsf{ff}, \mathsf{ffi}), \delta) \wedge \sim\mathsf{Happens}(\mathsf{ff}, \delta)\end{aligned} \tag{E10}$$

The meaning of axiom (E10) is that the obligation to do α is violated if the obligation that should hold at time δ does not appear. That is, a violation of obligation is triggered if the contents of the enforced obligation cannot be achieved by the deadline.

Assuming that the obligation in axiom (E9) is now violated. Based on the violation conditions set out in rule (LC5), we have the following axiom.

$$\begin{aligned}\mathsf{Happens}(\mathsf{Violation}(\mathsf{Oblig}(\mathsf{Principal}, \mathsf{PayInvoice}, \mathsf{ffi})), \delta) \longleftarrow \\ \mathsf{HoldsAt}(\mathsf{Oblig}(\mathsf{Principal}, \mathsf{PayInvoice}, \mathsf{ffi}), \delta) \wedge \sim\mathsf{Happens}(\mathsf{PayInvoice}, \delta)\end{aligned} \tag{E11}$$

From axiom (E11), we have $\mathsf{HoldsAt}(\mathsf{Oblig}(\mathsf{Principal}, \mathsf{PayInvoice}, \mathsf{ffi}), \delta)$ meaning that the principal has the obligation to pay the invoice by the deadline δ, i.e., 15 days from the issue date. If she did not pay on time, i.e., $\sim\mathsf{Happens}(\mathsf{PayInvoice}, \delta)$ becomes true, then a violation of the obligation will appear, and from axiom (E11) we derive $\mathsf{Happens}(\mathsf{Violation}(\mathsf{Oblig}(\mathsf{Principal}, \mathsf{PayInvoice}, \mathsf{ffi})), \delta)$.

Here, we have an issue regarding the temporal validity of the axiom above. In legal domain, a conclusion about whether a violation has occurred can only be made after the prescribed deadline has been passed. However, as can be seen in axiom (ax:axiom-12), the evaluation of the violation and the triggering event both take place at time instance δ, which is not consistent with our practice.

Consider, for example, the violation semantics of persistent obligations as illustrated in Figure 7, where the obligation o holds between times n and m. The obligation fluent can hold until the last moment and could be fulfilled by the deadline, i.e., m; where a violation can only be triggered if the obligation contents are not achieved after the deadline has passed, i.e., $m + 1$. According, there are other types of obligations, such as maintenance obligations, which are enforced in interval and the fluent for such obligations must hold for all instances of time in the interval. The violation of such obligations may occur anytime during the interval and even before the deadline. Here the objection with the PENELOPE semantics is on detecting violations at the deadline δ, which is meaningless for many real-life applications.

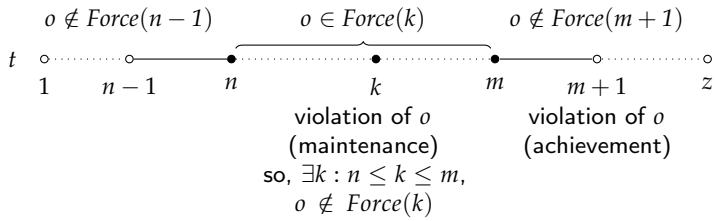

Figure 7. Violation semantics of persistent obligations (adopted from [11]).

Another issue with the violation semantics is due to the notion of Violations Without Reparation (VWR) that it built upon.

$$\begin{aligned}\mathrm{VWR}(\tau) : \{&\mathsf{Happens}(\mathsf{Violation}(\mathsf{Oblig}(\text{ß}, \mathsf{ff}, \mathsf{ffi})), \tau)\\ &\wedge \neg\exists \mathsf{Initiates}(\mathsf{Violation}(\mathsf{Oblig}(\text{ß}, \mathsf{ff}, \mathsf{ffi})), \mathsf{Oblig}(\text{ß}, \mathsf{ff}, \mathsf{ffi}), \tau)\}\end{aligned}$$

meaning that after the violation of an obligation happens, no further obligation will be initiated and the penalty can be imposed directly. Essentially, this notion of violation without reparation is closely related to Anderson's reduction view [52] of norms that whenever there is a violation, a direct penalty is imposed [53]. However, this may not be the case in legal domain. Instead, by compensating (or repairing) the violated obligation, the process can still be executed compliantly, but (only) in a sub-ideal situation.

Generally, the violation conditions of an obligation include the conditions of whether or not the obligation is compensable after violation. Such compensatory conditions were provided by analysts at the time of modeling the norms. The logical conditions in rule (LC6) is one such case where a sub-ideal situation is sought if the primary obligation is failed to be realised, which can be represented in PENELOPE as:

$$\begin{aligned}&\mathsf{Initiates}(\mathsf{Violation}(\mathsf{Oblig}(\mathsf{Principal}, \mathsf{PayInvoice}, \mathsf{ffi}'), \emptyset),\\ &\qquad \mathsf{Oblig}(\mathsf{Principal}, \mathsf{PayCompoundInvoice}, \delta), \tau) \wedge \delta' < \tau \leq \delta\end{aligned} \tag{E12}$$

$$\begin{aligned}&\mathsf{Initiates}(\mathsf{Violation}(\mathsf{Oblig}(\mathsf{Principal}, \mathsf{PayCompoundInvoice}, \mathsf{ffi}'), \emptyset),\\ &\qquad \mathsf{Oblig}(\mathsf{Principal}, \mathsf{PayCompoundInvoiceWith2.5\%Interest}, \delta), \tau) \wedge \delta' < \tau \leq \delta\end{aligned} \tag{E13}$$

Axiom (E12) states that the obligation to pay compound invoice (actual invoice + interest) enters into force at time instant τ after the primary obligation has been violated at a previous deadline δ'. Notice, EC does not lack expressiveness to represent the notion of compensation, which can be trivially modeled as shown above, but the notion of compensations is not considered in PENELOPE. However, we still have some issues with the above representation. First, because of the earlier discussed issues with the Initiates predicate, we may not get the effects of the newly initiated obligation onto the task. Also, it might be possible that the compensating obligation does not immediately enter into force but rather after some time delay because the compensating obligation may has its own specific triggering event. *Second*, the violation is triggered at the deadline not after the deadline has been passed. Nonetheless, at δ it might still be possible to pay the invoice since the obligation is still in force. Hence, using PENELOPE violation semantics triggering the violation at the deadline does not make sense because the representation does not reflect the actual situation.

As previously discussed (cf. Section 2), compensations are obligations themselves that can further be violated, and there can be situations where the compensated obligations are repetitively violated and compensated for. The logical conditions in rules (LC5) and (LC6) represent such cases where the repeated violations of the obligation to *PayInvoice* is compensated recursively in the sense that every time the obligation is violated, the contents of the violated obligation are accumulated to the compensatory obligation for a recursive compensation. In some situations, it might be possible that the primary obligation is not compensated in the first instance. For instance, consider the case when the Principal pays the 3% interest only but not the invoiced amount, the primary obligation is violated again. Then, in the next instance (axiom (E13)) the invoice is paid, the obligation is compensated, and no further action is necessary.

Even though the aim of compensating a violated obligation is to avoid any potential issues, not all violations can be compensated, even when penalties are enforced for non-compliance. For instance, the rule (LC6) prescribes the conditions for terminating the contract as follow.

$$\begin{aligned}\mathsf{Initiates}(\mathsf{Violation}(\mathsf{Oblig}(\mathsf{Principal}, \mathsf{PayCompoundInvoiceWith2.5\%Interest}, \mathsf{ffi}'), \emptyset),\\ \mathsf{Oblig}(\mathsf{Contractor}, \mathsf{TerminateContract}, \delta), \tau) \wedge \delta' < \tau \leq \delta\end{aligned} \tag{E14}$$

The meaning of the above axiom is that the contractor has an obligation to terminate the contract if the principal does not fulfil the obligation, compensating the violation. For this, we get $\mathsf{Violation}(\mathsf{Oblig}(\mathsf{Principal}, \mathsf{PayCompoundInvoiceWith2.5\%Interest}, \mathsf{ffi}'), \emptyset)$ meaning that the obligation to pay *compoundInvoice* is violated at time instant δ', which initiates the obligation *TerminateContract* at time τ to fulfill the obligation by a new deadline δ.

Accordingly, apart from the previously discussed issues with the predicates Initiates and Terminates, the conditions in this axiom give raise to an interesting question as to how to determine whether a process is complaint with the obligations after violations in PENELOPE. As the conditions in rule (LC6) (of clause (C6)) state, the contract is terminated after three violations of payments (and/or penalties). However, PENELOPE only provides support for violations detections, which is based on VWR. The language itself does not consider compensations; hence, issues may appear when modeling compensation obligations with PENELOPE.

5.2. Process Compliance Language (PCL)

PCL [9] is a formal language for the specifications of regulatory norms. The language is based on Defeasible Logic (DL) [54] and its modal extension (Modal Defeasible Logic (MDL) [55]), and can be used to model the deontic behaviours of a business contract. It consists of a finite set of (modal) literals that can be used to represent states of variables and/or tasks of a process. A literal is either an atomic proposition or its negation. That is, given a literal l, $\sim l$ denotes its complement. If $l = p$ then $\sim l = \neg p$, and if $l = \neg p$ then

$\sim l = p$. If X is a deontic operator and l is a literal, then Xl and its negation, X$\neg l$, are deontic literals. Preferences among (deontic) literals can be specified using a *superiority relation*, $>$.

Table 4 shows the set of deontic operators that PCL supports for representing different types of obligation, which includes almost all the deontic concepts that we have discussed in Section 2. In addition, PCL also supports CTD obligations through the notion of *reparation chain* or $\otimes$-expression, where $\otimes$ is the CTD structure operator [56]. That is, the meaning of an expression $c_1 \otimes c_2 \otimes \cdots \otimes c_n$, where $c_1, \ldots, c_n$ are (deontic) literals, is that c_1 is obligatory. But if the obligation of c_1 is violated, then the violation is compensated by c_2. Given that both c_1 and c_2 are violated, the obligation will then be compensated by c_3, and so on so forth.

Table 4. PCL obligation operators (adopted from [9]).

Operator	Obligation Type	Abbreviation
O^{m}	Maintenance	[OM]
$O^{a,\pi}_{pr}$	Achievement, Persistent, Preemptive	[OAPP]
$O^{a,\pi}_{n-pr}$	Achievement, Persistent, Non-preemptive	[OAPNP]
$O^{a,\tau}_{pr}$	Achievement, Non-persistent, preemptive	[OANPP]
$O^{a,\tau}_{n-pr}$	Achievement, Non-persistent, Non-preemptive	[OANPNP]
O^{p}	punctual	[P]

Modeling Contractual Constraints with PCL

In what follows, we are going to model the logical conditions of the contract presented in Section 3 using PCL, and discuss how interactions between different contractual clauses can be modeled efficiently.

Firstly, the prescribed constraints in rules (LC1) and (LC3) can be modeled as:

$$Paymentclaims \Longrightarrow_{[\mathsf{OAPNP}]} IssueInvoice \tag{P1}$$

$$Paymentclaims, IssuedInvoice, ReceivedInvoice \Longrightarrow_{[\mathsf{OAPNP}]} PayInvoicewithin15Days \tag{P2}$$

The meaning of formulas (P1) and (P2), respectively, is that given services provided, it is a persistent, non-preemptive obligation of the Provider to issue an invoice for making any payment claims, which must be consequently paid within 15 days from its issuance. Notice, we have events *IssuedInvoice* and *ReceivedInvoice* embedded into formula (P2) in order to capture the effects and dependencies between the agents giving thus a clear picture of the interaction.

As was previously discussed that handling the violations and dealing with obligations arising from the violations is a fundamental requirement for properly representing norms from a business process perspective. This is because generally processes operate in highly unpredictable environments and they tend to deviate from their intended course, and thus may not produce correct results; however, some norms prescribe counter measures on how to recover from the violation situations. The logical conditions in rules (LC4)–(LC6), respectively, present interesting situations where an obligation is violated and another obligation aiming to repair the violation enters into force, and if the violations is unrepairable then a penalty is enforced. PCL provides a non-boolean connectives $\otimes$ to form the expression connecting the primary obligation and its violation conditions with the compensating obligation. The PCL formulas for rules (LC4)–(LC6) can be written collectively as:

$$\begin{aligned} Paymentclaims, IssuedInvoice, ReceivedInvoice \Longrightarrow_{[\mathsf{OAPNP}]} & PayInvoicewithin15Days \\ & \otimes Paycompoundwith3\%Interest \otimes Paycompoundwith5.5\%Interest \end{aligned} \tag{P3}$$

$$\begin{aligned} Paymentclaims, IssuedInvoice, ReceivedInvoice \Longrightarrow_{[\mathsf{OAPNP}]} PayInvoicewithin15Days \\ \otimes Paycompoundwith3\%Interest \\ \otimes Paycompoundwith5.5\%Interest \otimes TerminateContract \end{aligned} \tag{P4}$$

Clearly, formula (P3) states that if the invoice is not paid within the stipulated time, a 3% interest is applicable, which in turn becomes 5.5% per day in case Principal fails to pay the invoice in next 7 days upon first violation. Essentially, in the chain expression, depending upon the situation, each violated obligation is compensated by another obligation amending the violation of the previous obligation.

Also, depending on an obligation and its violation conditions, an obligation can be recursively compensated for any violations. The PCL chain expression carries the information about the violated obligations in the formula, e.g., paying the invoice with 5.5% interest recursively compensates the primary obligation.

Since, the objective of compensation obligation is to achieve a sub-ideal situation after the violation, not in all cases the violation of an obligation and its compensation can continue indefinitely, thus, ultimate penalty is imposed [11]. Formula (P4) captures such situations where upon three consecutive defaults the contract is terminated. Notice that the logical conditions in formulas (P3) and (P4) can be individually modeled with the PCL trivially; however, it may introduce redundancies between the rules where the behaviour of the second rule may be implied by the first rule. PCL efficiently handles redundancies, and the reparation chains are created on the notion of subsumption which is out of the scope of this paper; see [9] for details.

$$ReceivedInvoice, UnresolvedConflict \Longrightarrow_{[\mathsf{Per}]} DelayPayment \tag{P5}$$

$$\begin{aligned} ReceivedInvoice, UnresolvedConflict, DelayedPayment, ResolvedConflict \\ \Longrightarrow_{[\mathsf{OAPNP}]} PayDelayedPayment \end{aligned} \tag{P6}$$

Formula (P5) models an exception as permission which explicitly permits the Principal to delay the payment until a resolution to an unresolved conflict is agreed (formula (P6)). Notice that, exception rules by no means compensate the violation, but they create situations where some non-ideal situations may be prevented temporarily. Accordingly, putting a temporary freeze on the obligation does not terminates the obligation; instead, it may prevent the effects that may be produced the task. Next, we model the norms for change request process for which the contract prescribes various requirements.

$$InForceContract \Longrightarrow_{[\mathsf{Per}]} RequestChange \tag{P7}$$

$$InForceContract \Longrightarrow_{[\mathsf{OAPNP}]} MakeFormalRequestforChanges \tag{P8}$$

The logical condition in rule (LC9) permits the Principal to request changes to the contractual services. Essentially, formula (P7) is a permission-based constraint which is mapped using PCL's semantics operator Per for permissions. Formula (P8) establishes that if the Principal intends to make a request for changes, then it must be formally requested for which a formal request for changes is required. The obligation resulting from this rule is a non-preemptive, persistent obligation. It is non-preemptive in the sense that the obligation only needs to be discharged if a formal request is sent.

$$\begin{aligned} InForceContract, FormalChangesRequest, MajorRequest \\ \Longrightarrow_{[\mathsf{Per}]} AcceptMajorRequest \end{aligned} \tag{P9}$$

$$\begin{aligned} InForceContract, FormalChangesRequest, MajorRequest \\ \Longrightarrow_{[\mathsf{Per}]} RejectMajorRequest \end{aligned} \tag{P10}$$

The meaning of formulas (P9) and (P10), respectively, is that given a formal request for major changes, the Provider is permitted to either accept or reject any major changes in the terms of the contract. Notice that we have modeled the rule separately into two independent rules, since, unlike PENELOPE which provide an XOR operator to model decision-based rules, PCL does not provide structural patterns (i.e., in this case, the OR operator) for such constraints. Besides, PCL does not consider *sort of*—meaning that a variable cannot be a literal and event simultaneously for creating formulas. Thus, it is required to divide them into multiple formulas, to distinguish the variables attached to the obligation as well as to determine the type of each obligation separately.

$$InForceContract, FormalChangesRequest, AcceptedMajorRequest \Longrightarrow_{[\mathsf{OAPNP}]} SeekApproval \tag{P11}$$

$$InForceContract, FormalChangesRequest, AcceptedMajorRequest, ApprovedChanges \Longrightarrow_{[\mathsf{OAPNP}]} PayCostsforMajorChanges \tag{P12}$$

In contrast, formula (P11) states that if the contractor accepts the changes request, then it is obligatory for the contractor to seek approval for the major changes. The obligation resulting from this rule is a persistent achievement obligation. As was previously discussed, a persistent obligation remains in force until the obligation has been fulfilled or a penalty is imposed; whereas formula (P12) states that the principal is responsible to pay the additional fee for any approved changes, i.e., $PayCostsforMajorChanges$ starts to hold.

$$InForceContract, FormalChangesRequest, MinorRequest \Longrightarrow_{[\mathsf{OM}]} \neg ClaimCostForChange \tag{P13}$$

$$InForceContract, FormalChangesRequest, ChangestoContract \Longrightarrow_{[\mathsf{OAPNP}]} DocumentChnages \tag{P14}$$

Formula (P13) maps an interesting case (cf. rule (LC15)) where the contractor is prevented from claiming any costs of minor changes, prescribing thus a prohibition. The formula captures the constraint as maintenance obligation because it is not possible to directly model prohibitions with the PCL semantics as the language does not provide any pattern (operator) for modeling prohibitions. Essentially, a maintenance obligation implements prohibitions [11] where the content of an obligation must be fulfilled for all the instances or intervals in which the obligation is in force. Hence, in modeling the prohibition in formula (P13) as maintenance obligation, we again subscribe to the duality relation between permission and obligation in mapping axiom (E7) in Section 5.1.

In contrast, formula (P14) is a non-preemptive persistent obligation where both the parties have an obligation to update the contract document, which can be trivially modeled using PCL.

6. Related Work

In this section we discuss existing studies where EC and DL (and its modal extension, i.e., MDL) have been used for modelling and reasoning about the legal norms. In addition, we also discuss comparative evaluations reported in literature and compare them with work presented in this paper.

6.1. Event Calculus-Based Modeling Approaches

EC has been extensively studied in multi-agent systems (MASs) for modeling and reasoning about agents behaviours, interaction, and planning [57–60], and has been equally exploited in studying the capabilities of modeling and reasoning with normative systems in legal and BPC domains.

In [61], a first-order model of using and representing norms for MASs is proposed. The norms have been considered from the events, event types and actions perspectives and formally modeled using EC. The framework also considers the deontic concepts (e.g.,

actions, agency and time) and constructs the formal representation using the fluent deontic and temporal predicates. In addition, the conflicts between the interactions among agents are captured using the violation predicates with respect to the time when an agent violates the interaction conditions.

Elakehal et al. [62] propose a model of self-managing MASs by adopting the EC-based axiomatisation approach proposed in [63], which allowed them to model the metric time constraints capturing the normative requirements in a formal way for the verification and run-time monitoring of the compliance constraints. The norms are modeled by means of goal/goal systems norms visually expressing the relations between the business process pairs to capture the business requirements. Paschke and Bichler [64] provide a logical framework for automating electronic contracts for representing complex business rules and business policies. The authors integrate the EC into other logical formalisms, e.g., Horn Logic, Deontic Logic, and Event-Conditions-Actions (ECA) rules, to model the contract states and deontic concepts (e.g., obligations, permissions) as time-varying fluents. Evans and Eyers [65] use EC for encoding deontic clauses in contracts for data use rules and monitoring of subsequent compliance of these rules. Their work is similar to the study of [66] with the exception that the logic programming used in their work is formalised in EC to represent the deontic states explicitly, while the latter is a declarative approach for modeling various contract rules types. Ours is different from these studies as we consider rather complex obligation modalities, and the effects of violations, while they only work on basic deontic notions, i.e., achievement, permissions and prohibitions, with the exception of the work by Elakehal et al. [62] where authors are able to model violations and obligations arising from the violation, yet the notion of perdurant obligations is not covered.

Yolum and Singh [67] study norms as *social commitments* capturing the obligations in the context of protocols. The authors employ Shanahan's version of EC [68] for modeling base-level and conditional commitments. Primarily, the study focuses on persistent commitments and provides EC axioms for reasoning commitments and operations on them. Also, it deals with the violation of commitments in the context of protocols to identify the non-compliant behaviour of an agent. Overall, the notion of commitments used in this work is somewhat similar to ours but their classes of commitments are context-specific, while our classes of obligations are generic and can be used in any context for compliance checking. Also, they do not have the notion of temporal proposition for modeling time interval which is imperative for modeling maintenance obligations.

Fornara and Colombetti [69] provide formal specifications of commitments and precommitments, institutionalised power, and context using the EC. The formal representation of norms is limited to obligations and permissions only, as in [8]. Also, no distinction between different types of obligations, and the effects of violation on the obligations has been made as we do; although the notion of sanctions has been formally presented in the study. A rather similar work by [70] provides EC-based formal specifications of obligations and permissions in the context of Ad-hoc Networks.

Alrawagfeh [71], on the other hand, proposes a norms representation approach using EC enabling the agents to use norms in their practical reasoning. The work considers only two classes of norms, i.e., obligations and prohibitions for which the authors introduced three fluents, namely: fPun and oPun, which refer to obligation norm violation and prohibition norm violation, respectively, and oRew for obligation fulfilment. The extended fluents can be used for representing norms that are composed of several actions combined with the norms' context. However, this work does not consider other types of norms and various obligation modalities as we do. Also, this study follows Anderson's reduction view of norms that suggests every violation of a norm is followed by a sanction [53]. We argue that initially, not in every case sanctions are/can be directly imposed as a sub-ideal situation can still make a business process compliant. The notion of compensation obligations and obligations perduring after the violation in our work are norms types that strengthen this argument.

6.2. Defeasible Logic-Based Modeling Approaches

Governatori and Milosevic [72] presented a formal system for describing the contracts in terms of deontic concepts of obligations, permissions and prohibitions. Their formalism supports the reasoning with violations, and laid the foundations for the contract specifications language, Business Contract Language (BCL). In order to capture the deontic behaviours of the contracts and their violations for compliance checking, the authors of [73] have extended BCL with a variant of MDL [55] and called it Formal Contract Language (FCL). Besides, the semantics of FCL also allow the determination of the current state of affairs (i.e., the notion of ideality) when comparing the business processes with the contractual constraints. However, these semantics support relatively simple normative expressions in which deontic constraints are expressed as single events; the support for rather complex event relationships is limited. In addition, the handling of deadlines in terms of FCL obligations modalities has been poorly expressed.

Milosevic et al. [74,75] use FCL to achieve compliance verification in a progressive manner. Initially, collaborative interactions or contract framing behaviour among all involving parties is identified. Then, the processes compliance within each contracting entity is determined. Followed by applying different heuristics at the end to reflect different contract conditions, specifying the set of actions to be taken when a violation occurs. The likelihood of contract violations is then checked at a supplementary stage of a process design. Scannapieco et al. [76], on the other hand, presented a formal approach using DL to integrate the business policy constraints with the organisational goals in such a way that a business process could fulfil the policy constraints and the organisational goals simultaneously.

However, as mentioned in [77], the proponents of defeasible reasoning also noticed that deontic logic based approaches might not be able to capture all the 8 fundamentals legal conceptions, namely: (i) *right* and *no right*, (ii) *privilege* and *duty*, (iii) *power* and *disability*, and (iv) *immunity* and *liability*, as stated in [78]. Recent work in legal representation includes two open standards proposed by OASIS, namely: OASIS LegalDocML (https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=legaldocml (accessed on: 15 October 2022)) and OASIS LegalRuleML (https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=legalruleml (accessed on: 15 October 2022)), and the translation from LegalRuleML to a variant of MDL has been proposed [79]. However, the concepts that can be represented in these emerging standards are still limited to the type of deontic behaviour that has been mentioned in Section 2 of this paper.

6.3. Other Approaches

Apart from EC and deontic logic-related formalisms, other formalisms, such as Linear Temporal Logic (LTL) and Branching Time Logic (BTL), and First Order Logic (FOL), have been proposed in business processes modeling. However, it seems that the LTL and BTL based frameworks (e.g., Business Process Modeling Notation-Query (BPMN-Q) [80,81], COMPAS [82], DECLARE [83,84]) are unable to model the majority of obligation types in a conceptually rich way and lack of support in modeling situations due to violations or CTD obligations. Besides, it has been proved formally that LTL cannot properly model permissions, which is an important issue for representing norms in real-life [85]. Similarly, the suitability of FOL-based frameworks (e.g., auditing framework [86], SeaFlows [87], eCRG [88]) is also questionable. As discussed in [89], it has been argued that FOL is not suitable to model real-life norms since FOL has no conceptual relevance to the legal domain.

In addition, there are also formalisms that have been developed to resolve issues in the norms modelling process. For instance, Elgammal et al. [90] has developed a language, Compliance Request Language (CRL), to model organization-specific compliance requirement. In a similar vein, González and Delgado [91] have recently proposed a language, Compliance Requirement Modeling Language (CRML), to model compliance requirements over BPMN. However, the scope of the former is a bit limited and can only be used as an auxiliary step to represent internalized compliance requirements into

formal language, such as LTL; whereas the latter was limited in modeling compliance requirements in inter-organizational collaborative (communication) processes, lacking the support to model other aspects of compliance requirements such as human resources, equipments, dataset, etc. Besides, no details have been provided as to how this language is being implemented.

6.4. Works on Comparative Analysis

In the context of COMPAS project, Elgammal et al. [82] performed a comparative analysis of three modeling languages from a family of two logics for a design-time compliance request language for modeling the compliance requirements. The performed analysis is based on the 11 key features identified using two case studies. These features also include the formality and expressiveness of the compared languages. The authors used a subset of norms types, thus, it is hard to establish the faithfulness of their perform comparison.

Ly et al. [92] proposed an evaluation framework for comparing the core functionalities of existing compliance monitoring approaches. Their framework is built on typical ten compliance monitoring functionalities derived from literature and case studies from different domains. However, their evaluation is limited in scope as their framework is restricted only to compliance monitoring approaches, while other approaches have not been considered. In addition, the core functionality of a compliance rule language, i.e., the expressive power to model the compliance rules, has not been incorporated into their analysis.

Novotná and Libal [93] provided correctness, transparency, comprehensibility, and multiple interpretations support as the criteria for good formalisation of legal rules. Using these properties, they proposed an evaluation methodology for evaluating systems, tools and method for formalising legal rules. Wang et al. [94], on the other hand, reported an empirical analysis of model understanding and evaluated a range of business process and rule integration approaches from a rule linking perspective. These studies are different from our evaluations as the former does not consider expressiveness of the compliance rule modelling to intuitive capture the meanings of the legal rules. Whereas latter focuses on gaining an understanding on the performance of business processes when linking the legal rules in terms of understanding effectiveness, efficiency, perceived mental efforts and visual attention when modelling and linking the legal rules with business processes.

A rather similar approach has been proposed in [91]; however, the focus their is limited only to the compliance requirement for a modelling language. In comparison to our work, the authors explore the connection between the compliance modelling language and with the elements of BPMN 2.0. In addition, to this they present compliance requirements model specific for specific business process, and initial view on the post-more compliance evaluation with process mining. Whereas, we focus on the expressiveness of compliance modelling language to intuitively capture the meaning of legal rule.

Fenech et al. [95] evaluated the expressiveness of Computation Tree Logic (CTL) and LTL with a deontic-based contract language $\mathcal{CL}$ [96] and operation models (CSP [97]) for modeling the full specifications of electronic contracts. In contrast, we evaluated the formal specifications of the business contract in terms of obligation modalities by defining the different types of obligations and examined whether or not the intuition of these modalities could be captured using different formalisms. Our evaluation differ from these studies as the former evaluations consider simple achievements, permission and prohibition constraints only examining the formality and expressiveness of the considered formalisms, whereas the latter focuses on the specifications of functional and behaviour requirements only.

7. Discussion

In Section 5, we have formally evaluated the semantics of PENELOPE and PCL in modeling different types of norms (cf. Section 2) under different conditions, and a number of issues were identified. Table 5 provides a snapshot of the characteristics and the features supported by the two frameworks.

Table 5. PENELOPE vs. PCL.

	PENELOPE	PCL
Formalism base	Event Calculus	Defeasible Logic
Complexity	Polynomial	Linear
Features Supported [a]		
Deontic Properties		
Obligation	✓̸	✓
Prohibition	✓̸	✓
Permission	✓	✓
CTD structure	✗	✓

[a] Supported: ✓ Partially supported: ✓̸ Not supported: ✗.

We have identified that PENELOPE is unable to model all types of obligations, such as preemptive, achievement, and maintenance obligations, and conditions that are associated with them, which is mainly caused by the restrictions of the EC predicates used in PENELOPE. Essentially, the EC's base predicates Initiates and HoldsAt are unable to capture correctly when an obligation enters into force, meaning that these predicates (and their dependencies) may not be actually, or accurately, triggered at the correct time points.

Accordingly, in many situations, an obligation may have its own set of triggers, and firing one of these triggers will initiate the obligation. However, with the current Initiates predicate it is not possible to have this effect; thus, checking the compliance of the business process will be meaningless. In a similar vein, the Terminates predicate also inflicts a similar issue as an obligation can have a validity period until when it is in force and after which it is terminated (upon fulfilment), or a violation is triggered. As the obligations cannot be terminated at the correct time points, compliance issues may arise due to the early (or later) termination of the obligations. Contrary to PENELOPE, PCL does not have these issues as the language itself is able to capture the meaning of these notions through the use of different modal operators [9].

Similarly, prohibitions guide agents behaviour by putting constraints on what agents must refrain from performing during the course of their interaction in a task [34]. In PENELOPE, prohibitions are considered *only* under *closed world assumption* [48] and implicitly assumed when no obligations or permissions can be derived. With existing EC predicates, a prohibition can be modeled as a negative permission; however, the problem with capturing the time when a deadline happens remains the same with the Initiates predicate. Hence, the PENELOPE's violation semantics are unable to initiative capture the violation (see, Section 5.1) as it is not possible to signal the deadline until which an obligation conditions can be fulfilled; otherwise, a violation may be triggered.

In contrast, PCL represents prohibitions as maintenance obligations and has provided syntax to repair, or compensate, breached obligations. PENELOPE, however, does not include this feature, and users are required to model the compensations as events and embed them into the violation events to mimic the effects.

As far as perdurance obligations are concerned, the current variants of both PENELOPE and PCL are not able to represent perdurant obligations [11]. However, this issue has been addressed in [98] enabling PCL to effectively capture all types of normative requirements.

8. Final Remarks

Organisations are automating their business operations, including the fulfilment of their compliance reporting requirements. For this purpose, they use customisable tools to automatically generate compliance reports and other related documents. However, the effectiveness of such tools depends largely on their capabilities of providing modeling and

reasoning support to different types of norms, and on how accurate the legal/normative requirements is in the modeled norms.

In the past decades, many promising research efforts in this area have been reported in the literature. For instance, Breaux et al. [99,100] propose a tabular-format approach for acquiring and representing data access requirements, and the management of the priorities between the data and access requirements. Furthermore, Sapkota et al. [101] identify the document structure and exploit semantic concepts in the process ontology for the extraction of regulations to facilitate the automated alignment of the regulations. The former methodology primarily focused on the constraints related to the data and the objects of interest, and do not incorporate other process aspects. Moreover, their document analysis also limited the rule record format. The methodology reported here is complementary but different from these studies as we incorporate the natural IF... THEN structure to analyse the contents of the legal document and to extract the activities and types of obligations and their effects on the interactions between the activities. In addition to that, the IF... THEN structure can also incorporate different process aspects, such as control-flow, data, resources, etc, which are largely ignored during the analysis of legal documents. Using a real-life example, we have shown that the proposed methodology provides the guidance on *how to abstract the structure of the regulatory texts* to determine, which rules are relevant to an activity(-ies) of a business process under what conditions (effects of the norms), and what is the specific type of the rules. Moreover, our methodology also facilitates the backtracking of the interactions between the propositions of the rules as well as dependencies between them, and thus improves the traceability of the legal norms. Besides, the formal nature of the IF... THEN structure also allows us to directly embed the information extracted into a logical formalism for automated compliance checking.

Mining normative information from legal documents is a time-consuming, error-prone, and technically complex task that even human lawyers have difficulties understanding and applying [102]. Thus, being able to capture the real intention of these information and formalise them precisely in a machine-understandable formalism is imperative for automated compliance checking [103,104]. In this regard, the expressiveness of the underlying formalism used in a particular framework is of vital importance to achieve success. In this paper, we have evaluated two prominent modeling languages, namely, PENELOPE and PCL, that can be used to model and validate business processes during design-time. Our objective was to formally examine *how these languages model* and *reason* about normative requirements. We started by discussing our previously proposed classification model and the formal semantics of each of the classes. The classification is based on the temporal validity of norms, what constitutes a violation, and the effects of the violations. Then, using a real life example, we modeled various contractual conditions with PENELOPE and PCL. Our formal evaluation showed that PENELOPE has fundamental deficiencies in modeling all types of norms for compliance checking. These deficiencies are not specific to the formal logic EC which has some issues with its base predicates Initiates and HoldsAt (due to which the effects of the tasks cannot be captured onto business processes) but also the notions that are not considered by the language, such as prohibitions. The authors in [34] tackled some deficiencies in PENELOPE and introduced permission P and forbidden F deontic operators to properly capture obligations and permissions, respectively.

As far as PCL is concerned, it can model almost all obligation classes under the classification model. However, at the moment, the reparation chain operator $\otimes$ is unable to model recursive compensation of multiple violations of an obligation (or compensation obligation). Besides, legal documents may include nested modalities prescribing obligations, permissions or exceptions [21], a non-monotonic treatment and handling of nested modalities per se is a difficult task [105,106]. PCL is unable to represent nested modalities. Besides, the evaluated version of PCL cannot model perdurant obligations, which has been addressed elsewhere in [98].

As the future work, we plan to continue these evaluations and compare other CMFs based on different formalisms, (such as LTL, BTL, FOL, etc.), graph-based (such as BPMN-

Q) and primitive based languages to enhance the generalisability of the evaluation. Besides, as was mentioned before that compliance requirements might be relevant to different aspects of business processes, we also plan to include them in our future evaluations.

Another line of interest for future work is evaluating the prescriptive safety standards that cover a ranged of software life-cycle activities. In particular, we are interested in evaluating process models prescribed in automotive (ISO-26262: https://www.iso.org/standard/68383.html (accessed on: 15 October 2022)), railway (CENLEC EN 50128: https://standards.globalspec.com/std/14317747/ (accessed on: 15 October 2022)), medical devices (DO-178C: https://www.adacore.com/industries/avionics/what-is-do-178c (accessed on: 15 October 2022)), and safety-critical systems (IEC 61508: https://webstore.iec.ch/publication/5515 (accessed on: 15 October 2022)), as these standards present complex requirements for the management of software development activities.

Author Contributions: Conceptualization, M.H. and H.-P.L.; methodology, M.H. and H.-P.L.; investigation, M.H. and H.-P.L.; resources, M.H. and H.-P.L.; writing—original draft preparation, M.H. and H.-P.L.; writing—review and editing, M.H. and H.-P.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
BCL	Business Contract Language
BPMN	Business Process Modeling Notation
BPMN-Q	Business Process Modeling Notation-Query
BTL	Branching Time Logic
CbD	Compliance by Design
CMF	Compliance Management Framework
CRL	Compliance Request Language
CTD	Contrary-to-Duty
CTL	Computation Tree logic
DL	Defeasible Logic
EC	Event Calculus
ECA	Event-Conditions-Actions
FCL	Formal Contract Language
FOL	First Order Logic
LTL	Linear Temporal Logic
MAS	Multi-Agent System
MDL	Modal Defeasible Logic
PCL	Process Compliance Language
PENELOPE	Process Entailment for Elicitation of Obligations and Permissions
VWR	Violations Without Reparation

References

1. US Government. *Public Company Accountng Reforms and Investor Protection Act (Sarbanes-Oxley Act)*; Public Law 107-204, 116 Stat. 745; US Government: Washington, DC, USA, 2002.
2. Basel Committee on Banking Supervision. *Basel III: The Liquidity Coverage Ratio and Liquidity Risk Monitoring Tools*; Banks for International Settlements: Basel, Switzerland, 2013.
3. Hashmi, M.; Governatori, G. Norms modeling constructs of business process compliance management frameworks: A conceptual evaluation. *Artif. Intell. Law* **2017**, *26*, 251–305. [CrossRef]
4. Lohmann, N. Compliance by Design for Artifact-centric Business Processes. *Inf. Syst.* **2012**, *38*, 606–618. [CrossRef]
5. Hashmi, M.; Governatori, G.; Lam, H.P.; Wynn, M.T. Are We Done with Business Process Compliance: State-of-the-Art and Challenge Ahead. *Knowl. Inf. Syst.* **2018**, *57*, 79–133. [CrossRef]
6. Becker, J.; Delfmann, P.; Eggert, M.; Schwittay, S. Generalizability and Applicability of Model-Based Business Process Compliance-Checking Approaches—A State-of-the-Art Analysis and Research Roadmap. *Bus. Res.* **2012**, *5*, 221–247. [CrossRef]
7. Lee, A.S.; Baskerville, R.L. Generalizing Generalizability in Information Systems Research. *Inf. Syst. Res.* **2003**, *14*, 221–243. [CrossRef]
8. Goedertier, S.; Vanthienen, J. Designing Compliant Business Processes with Obligations and Permissions. In *Business Process Management Workshops 2006*; Eder, J., Dustdar, S., Eds.; Springer: Vienna, Austria, 2006; pp. 5–14. [CrossRef]
9. Governatori, G.; Rotolo, A. A Conceptually Rich Model of Business Process Compliance. In Proceedings of the 7th Asia-Pacific Conference on Conceptual Modelling, APCCM'10, Brisbane, Australia, 18–21 January 2010; Australian Computer Society, Inc.: Brisbane, Australia, 2010; pp. 3–12.
10. Hashmi, M.; Governatori, G.; Wynn, M.T. Normative Requirements for Business Process Compliance. In Proceedings of the 3rd Australasian Symposium on Service Research and Innovation, ASSRI'13, Sydney, NSW, Australia, 27–29 November 2013; Springer: Sydney, Australia, 2013; pp. 100–116.
11. Hashmi, M.; Governatori, G.; Wynn, M. Normative Requirements for Regulatory Compliance: An Abstract Formal Framework. *Inf. Syst. Front.* **2016**, *18*, 429–455. [CrossRef]
12. Kant, I. *Groundwork for the Metaphysics of Morals*; Cambridge University Press: Cambridge, UK, 2002.
13. Kant, I. *Kant: The Metaphysics of Morals*; Gregor, M., Translator; Cambridge University Press: Cambridge, UK, 2003.
14. Mill, J.S. *Utilitarianism*, Translated Version ed.; Oxford University Press: Oxford, UK, 1998.
15. Boella, G.; van der Torre, L. Regulative and Constitutive Norms in Normative Multiagent Systems. In Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning, KR'04, Haifa, Israel, 31 July–5 August 2004; AAAI Press: Whistler, BC, Canada, 2004; pp. 255–265.
16. Boella, G.; van der Torre, L. Permissions and Obligations in Hierarchical Normative Systems. In Proceedings of the 9th International Conference on Artificial Intelligence and Law, ICAIL '03, Scotland, UK, 24–28 June 2003; ACM: Scotland, UK, 2003; pp. 109–118.
17. Brown, M.A. Conditional obligation and positive permission for agents in time. *Nord. J. Philos. Log.* **2000**, *5*, 83–111. [CrossRef]
18. Makinson, D.; van der Torre, L. Permission from an Input/Output Perspective. *J. Philos. Log.* **2003**, *32*, 391–416. [CrossRef]
19. Boella, G.; van der Torre, L. Permissions and Undercutters. In Proceedings of the 5th International Workshop Nonmonotonic Reasoning, Action and Change, Acapulco, Mexico, 10–11 August 2003; NRAC: Acapulco, Mexico, 2003.
20. Stolpe, A. A theory of permission based on the notion of derogation. *J. Appl. Log.* **2010**, *8*, 97–113. [CrossRef]
21. von Wright, G.H. *Norm and Action: A Logical Enquiry*; Routledge and Kegan Paul: London, UK, 1963.
22. Alchourrón, C.; Bulygin, E. The Expressive Conception of Norms. In *New Studies in Deontic Logic*; Hilpinen, R., Ed.; Springer: Berlin/Heidelberg, Germany; Synthese Library: Dordrecht, The Netherlands, 1981; Volume 152, pp. 95–124.
23. Opalek, K.; Woleński, J. Normative Systems, Permission and Deontic Logic. *Ratio Juris* **1991**, *4*, 334–348. [CrossRef]
24. Weinberger, O. The expressive conception of norms—An impasse for the logic of norms. *Law Philos.* **1985**, *4*, 165–198. [CrossRef]
25. Wang, L. Strong Permission in Prescriptive Causal Models. In *New Frontiers in Artificial Intelligence*; Otake, M., Kurahashi, S., Ota, Y., Satoh, K., Bekki, D., Eds.; Springer International Publishing: Kanagawa, Japan, 2017.
26. Governatori, G. Thou Shalt is not You Will. In Proceedings of the 15th International Conference on Artificial Intelligence and Law, ICAIL '15, San Diego, CA, USA, 8–12 June 2015; Atkinson, K., Ed.; ACM: San Diego, CA, USA, 2015; pp. 63–68.
27. Hart, H.L.A. Are There Any Natural Rights? *Philos. Rev.* **1955**, *64*, 175–191.
28. Hart, L.A. H. L. A. Hart on Legal and Moral Obligation. *Mich. Law Rev.* **1974**, *73*, 443–458.
29. Brandt, R.B. V.—The Concepts of Obligations and Duty. *Mind* **1964**, *LXXIII*, 374–393. [CrossRef]
30. Dubbink, W. Duty. In *The SAGE Encyclopedia of Business Ethics and Society*, 2nd ed.; Kolb, R.W., Ed.; SAGE: Thousand Oaks, CA, USA, 2018; Volume 2, pp. 994–1003.
31. Herman, B. On the Value of Acting from the Motive of Duty. *Philos. Rev.* **1981**, *90*, 359.
32. Verweij, M. The Concept of Duty and Obligation. In *Preventive Medicine between Obligation and Aspiration*; Springer: Dordrecht, The Netherlands, 2000; pp. 103–122. [CrossRef]
33. Hill, T.E. 8. Kant on Imperfect Duty and Supererogation. In *Dignity and Practical Reason in Kant's Moral Theory*; Cornell University Press: Ithaca, NY, USA, 2019; pp. 147–175. [CrossRef]

34. Hashmi, M.; Governatori, G.; Wynn, M.T. Modeling Obligations with Event-Calculus. In Proceedings of the 8th International Workshop on Rules and Rule Markup Languages for the Semantic Web, RuleML 2014, Prague, Czech Republic, 8–22 April 2014; Springer: Prague, Czech Republic, 2014; pp. 296–310.
35. Chisholm, R.M. Contrary-to-Duty Imparatives and Deontic Loigc. *Analysis* **1963**, *24*, 33–36.
36. Prakken, H.; Sergot, M. Contrary-to-duty obligations. *Stud. Log.* **1996**, *57*, 91–115. [CrossRef]
37. Bassam, S.; Gardiner, D.; Sheridan, H. (Eds.) *The Law Handbook: Your Practical Guide to the Law in NSW*, 2017th ed.; Redfern Legal Centre Publishing: NSW, Australia 2018.
38. Breaux, T.D.; Antón, A.I. A Systematic Method for Acquiring Regulatory Requirements: A Frame-Based Approach. In Proceedings of the 6th International Workshop on Requirements for High Assurance Systems, RHAS-6, Kebbutz Sehfayim, Israel, 4–6 July 2006; Software Engineering Institute: Pittsburg, PA, USA, 2006.
39. Hashmi, M. A Methodology for Extracting Legal Norms from Regulatory Documents. In Proceedings of the 2015 IEEE 19th International Enterprise Distributed Object Computing Workshop, EDOCW 2015, Adelaide, Australia, 22–25 September 2015; IEEE: Adelaide, SA, Australia, 2015; pp. 41–50.
40. King, T.C.; De Vos, M.; Dignum, V.; Jonker, C.M.; Li, T.; Padget, J.; van Riemsdijk, M.B. Automated multi-level governance compliance checking. *Auton. Agents Multi-Agent Syst.* **2017**, *31*, 1283–1343. [CrossRef]
41. Gordon, T.F.; Governatori, G.; Rotolo, A. Rules and Norms: Requirements for Rule Interchange Languages in the Legal Domain. In *International Symposium on Rule Interchange and Applications, RuleML 2009*; Governatori, G., Hall, J., Paschke, A., Eds.; Springer: Berlin/Heidelberg, Germany; Las Vegas, NV, USA, 2009; pp. 282–296.
42. Sartor, G. Legal Reasoning: A Cognitive Approach to the Law. In *A Treatise of Legal Philosophy and General Jurisprudence*; Roversi, C., Ed.; Springer: Dordrecht, The Netherlands, 2005; Volume 5.
43. Makinson, D. On a Fundamental Problem of Deontic Logic. In *Norms, Logics and Information Systems. New Studies in Deontic Logic and Computer Science*; Frontiers in Artificial Intelligence and Applications; McNamara, P., Prakken, H., Eds.; IOS Press: Amsterdam, The Netherlands, 1999; Volume 49, pp. 29–53.
44. Hashmi, M.; Governatori, G. A Methodological Evaluation of Business Process Compliance Management Frameworks. In Proceedings of the 1st Asia Pacific Conference on Business Process Management, AP-BPM 2013, Beijing, China, 29–30 August 2013; Song, M., Wynn, M.T., Liu, J., Eds.; Springer: Beijing, China, 2013; pp. 106–115. [CrossRef]
45. Kowalski, R.; Sergot, M. A Logic-Based Calculus of Events. In *Foundations of Knowledge Base Management*; Topics in Information Systems; Schmidt, J., Thanos, C., Eds.; Springer: Berlin/Heidelberg, Germany, 1989; pp. 23–55.
46. Miller, R.; Shanahan, M. The Event Calculus in Classical Logic—Alternative Axiomatisations. *Electron. Trans. Artif. Intell.* **1999**, *4*, 77–105.
47. Miller, R.; Shanahan, M. Some Alternative Formulations of the Event-Calculus. In *Computational Logic: Logic Programming and Beyond: Essays in Honour of Robert A. Kowalski Part II*; Kakas, A.C., Sadri, F., Eds.; Springer: Berlin/Heidelberg, Germany, 2002; pp. 452–490.
48. Kunen, K. Negation in logic programming. *J. Log. Program.* **1987**, *4*, 289–308. [CrossRef]
49. Hansen, J.; Pigozzi, G.; van der Torre, L.W.N. Ten Philosophical Problems in Deontic Logic. In *Normative Multi-Agent Systems*; Dagstuhl Seminar Proceedings (DagSemProc); Boella, G., van der Torre, L., Verhagen, H., Eds.; Schloss Dagstuhl—Leibniz-Zentrum für Informatik: Dagstuhl, Germany, 2007; Volume 7122.
50. Alchourrón, C.E. Philosophical Foundations of Deontic Logic and the Logic of Defeasible Conditionals. In *Deontic Logic in Computer Science*; Meyer, J.J.C., Wieringa, R.J., Eds.; John Wiley & Sons, Inc.: New York, NY, USA, 1994; pp. 43–84.
51. Awad, A. A Compliance Management Framework for Business Process Models. Ph.D. Thesis, Hasso Plattner Institute, University of Potsdam, Potsdam, Germany, 2010.
52. Anderson, A.R. A Reduction of Deontic Logic to Alethic Modal Logic. *Mind* **1958**, *67*, 100–103. [CrossRef]
53. Soeteman, A. (Ed.) *Pluralism and Law*; Springer: Dordrecht, The Netherlands, 2001; Volume 4. [CrossRef]
54. Nute, D. Defeasible logic: Theory, Implementation and Applications. In Proceedings of the 14th International Conference on Applications of Prolog, INAP 2001, Tokyo, Japan, 20–22 October 2001; Springer: Tokyo, Japan, 2001; pp. 151–169.
55. Governatori, G.; Rotolo, A. BIO Logical Agents: Norms, Beliefs, Intentions in Defeasible Logic. *Auton. Agents Multi-Agent Syst.* **2008**, *17*, 36–69. [CrossRef]
56. Governatori, G.; Rotolo, A. Logic of Violations: A Gentzen System for Reasoning with Contrary-To-Duty Obligation. *Australas. J. Log.* **2006**, *4*, 193–215.
57. Alberti, M.; Gavanelli, M.; Lamma, E.; Chesani, F.; Mello, P.; Torroni, P. Compliance Verification of Agent Interaction: A Logic-based Software Tool. *Appl. Artif. Intell.* **2006**, *20*, 133–157.
58. Chesani, F.; Mello, P.; Montali, M.; Torroni, P. Representing and Monitoring Social Commitments using the Event Calculus. *Auton. Agents Multi-Agent Syst.* **2013**, *27*, 85–130. [CrossRef]
59. Flores, R.A.; Kremer, R.C. To Commit or Not to Commit: Modeling Agent Conversations for Action. *Comput. Intell.* **2002**, *18*, 120–173. [CrossRef]
60. Yolum, P.; Singh, M.P. Flexible Protocol Specification and Execution: Applying Event-Calculus Planning using Commitments. In Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS '02, Bologna, Italy, 15–19 July 2002; ACM: Bologna, Italy, 2002; pp. 527–534.

61. Stratulat, T.; Clérin-Debart, F.; Enjalbert, P. Norms and Time in Agent-Based Systems. In Proceedings of the 8th International Conference on Artificial Intelligence and Law, ICAIL'01, St. Louis, MO, USA, 21–25 May 2001; ACM: St. Louis, Missouri, USA, 2001; pp. 178–187.
62. Elakehal, E.E.; Marco, M.; Julian, P. Run-time Verification of MSMAS Norms Using Event-Calculus. In Proceedings of the 1st International Workshop on Quality Assurance for Self-Adaptive, Self-Organising Systems, QA4SASO, London, UK, 8–12 September 2014; IEEE: London, UK, 2014; pp. 110–115. [CrossRef]
63. Chesani, F.; Mello, P.; Montali, M.; Torroni, P. Verification of Choreographies During Execution Using the Reactive Event Calculus. In Proceedings of the 5th International Workshop on Web Services and Formal Methods, WS-FM 2008, Milan, Italy, 4–5 September 2008; Bruni, R., Wolf, K., Eds.; Springer: Berlin/Heidelberg, Germany; Milan, Italy, 2009; pp. 55–72.
64. Paschke, A.; Bichler, M. SLA Representation, Management and Enforcement. In Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service, EEE'05, Hong Kong, China, 29 March–1 April 2005; IEEE: Hong Kong, China, 2005; pp. 158–163.
65. Evans, D.; Eyers, D.M. Deontic Logic for Modelling Data Flow and Use Compliance. In Proceedings of the 6th International Workshop on Middleware for Pervasive and Ad-hoc Computing, MPAC '08, Leuven, Belgium, 1–5 December 2008; ACM: Leuven, Belgium, 2008; pp. 19–24.
66. Grosof, B.N.; Labrou, Y.; Chan, H.Y. A Declarative Approach to Business Rules in Contracts: Courteous Logic Programs in XML. In Proceedings of the 1st ACM Conference on Electronic Commerce, EC '99, Denver, CO, USA, 3–5 November 1999; ACM: Denver, CO, USA, 1999; pp. 68–77. [CrossRef]
67. Yolum, P.; Singh, M.P. Reasoning about Commitments in the Event Calculus: An Approach for Specifying and Executing Protocols. *Ann. Math. Artif. Intell.* **2004**, *42*, 227–253. [CrossRef]
68. Shanahan, M. *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*; MIT Press: Cambridge, MA, USA, 1997.
69. Fornara, N.; Colombetti, M. Specifying Artificial Institutions in the Event Calculus. In *Handbook of Research on Multi-Agent Systems: Sematnics and Dynamics of Organisational Models*; IGI Global: Hershey, PA, USA, 2009; pp. 335–366. [CrossRef]
70. Artikis, A.; Kamara, L.; Pitt, J.; Sergot, M. A Protocol for Resource Sharing in Norm-Governed Ad Hoc Networks. In Proceedings of the 2nd International Workshop on Declarative Agent Languages and Technologies II, DALT 2004, New York, NY, USA, 19 July 2004; Leite, J., Omicini, A., Torroni, P.,Yolum, P., Eds.; Springer: New York, NY, USA, 2005; pp. 221–238. [CrossRef]
71. Alrawagfeh, W. Norm Representation and Reasoning: A Formalization in Event Calculus. In Proceedings of the International Conference on Principles and Practice of Multi-Agent Systems, PRIMA 2013, Dunedin, New Zealand, 1–6 December 2013; Boella, G., Elkind, E., Savarimuthu, B.T.R., Dignum, F., Purvis, M.K., Eds.; Springer: Dunedin, New Zealand, 2013; pp. 5–20. [CrossRef]
72. Governatori, G.; Milosevic, Z. Dealing with Contract Violations: Formalism and Domain Specific Language. In Proceedings of the 9th International Enterprise Distributed Object Computing Conference, EDOC 2005, Enschede, The Netherlands, 19–23 September 2005; IEEE Computer Society: Enschede, The Netherlands, 2005; pp. 46–57.
73. Governatori, G.; Milosevic, Z.; Sadiq, S. Compliance Checking between Business Processes and Business Contracts. In Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2006, Hong Kong, China, 16–20 October 2006; IEEE Computing Society: Hong Kong, China, 2006; pp. 221–232. [CrossRef]
74. Milosevic, Z.; Sadiq, S.; Orlowska, M. Towards a Methodology for Deriving Contract-Compliant Business Processes. In Proceedings of the 4th International Conference on Business Process Management, BPM 2006, Vienna, Austria, 5–7 September 2006; Dustdar, S., Fiadeiro, J., Sheth, A., Eds.; Springer: Vienna, Austria, 2006; pp. 395–400.
75. Milosevic, Z.; Sadiq, S.; Orlowska, M. Translating Business Contract into Compliant Business Processes. In Proceedings of the 10th IEEE International on Enterprise Distributed Object Computing Conference, EDOC '06, Hong Kong, China, 16–20 October 2006; IEEE: Hong Kong, China, 2006; pp. 211–220.
76. Scannapieco, S.; Governatori, G.; Olivieri, F.; Cristani, M. Designing for Compliance: Norms and Goal. In Proceedings of the 5th International Symposium on Rule-Based Modeling and Computing on the Semantic Web, RuleML 2011, Ft. Lauderdale, FL, USA, 3–5 November 2011; Olken, F., Palmirani, M., Sottara, D., Eds.; Springer: Ft Lauderdale, FL, USA, 2011; pp. 282–297. [CrossRef]
77. Otto, P.N.; Antón, A.I. Addressing Legal Requirements in Requirements Engineering. In Proceedings of the 15th IEEE International Requirements Engineering Conference, RE'07, New Delhi, India, 15–19 October 2007; IEEE: Delhi, India, 2007; pp. 5–14. [CrossRef]
78. Hohfeld, W.N. Some Fundamental Legal Conception as Applied in Judicial Reasoning. *Yale Law J.* **1919**, *23*, 16.
79. Lam, H.P.; Hashmi, M. Enabling Reasoning with LegalRuleML. *Theory Pract. Log. Program.* **2019**, *19*, 1–26. [CrossRef]
80. Awad, A.; Weidlich, M.; Weske, M. Specification, Verification and Explanation of Violation for Data Aware Compliance Rules. In Proceedings of the 7th International Joint Conference on Service-Oriented Computing, ICSOC 2009, Stockholm, Sweden, 24–27 November 2009; Baresi, L., Chi, C.H., Suzuki, J., Eds.; Springer: Berlin/Heidelberg, Germany; Stockholm, Sweden, 2009; pp. 500–515. [CrossRef]
81. Awad, A.; Weidlich, M.; Weske, M. Visually Specifying Compliance Rules and Explaining their Violations for Business Processes. *J. Vis. Lang. Comput.* **2011**, *22*, 30–55. [CrossRef]
82. Elgammal, A.; Turetken, O.; van den Heuvel, W.J.; Papazoglou, M. On the Formal Specification of Regulatory Compliance: A Comparative Analysis. In Proceedings of the ICSOC 2010 International Workshops, San Francisco, CA, USA, 7–10 December 2010; Maximilien, E.M., Rossi, G., Yuan, S.T., Ludwig, H., Fantinato, M., Eds.; Springer: San Francisco, CA, USA, 2011; pp. 27–38.

83. Pesic, M.; van der Aalst, W. A Declarative Approach for Flexible Business Processes Management. In *Business Process Management Workshops 2006*; Eder, J., Dustdar, S., Eds.; Springer: Vienna, Austria, 2006; pp. 169–180. [CrossRef]
84. Pesic, M.; Schonenberg, H.; van der Aalst, W. DECLARE: Full Support for Loosely-Structured Processes. In Proceedings of the 11th IEEE International Conference on Enterprise Distributed Object Computing, EDOC 2007, Annapolis, MD, USA, 15–19 October 2007; pp. 287–298. [CrossRef]
85. Governatori, G.; Hashmi, M. No Time for Compliance. In Proceedings of the 19th IEEE The Enterprise Computing Conference, EDOC'15, Adelaide, Australia, 21–25 September 2015; IEEE: Adelaide, SA, Australia, 2015; pp. 9–18.
86. Ghose, A.; Koliadis, G. Auditing Business Process Compliance. In Proceedings of the 5th International Conference on Service-Oriented Computing, ICSOC 2007, Vienna, Austria, 17–20 September 2007; Krämer, B.J., Lin, K.J., Narasimhan, P., Eds.; Springer: Berlin/Heidelberg, Germany; Vienna, Austria, 2007; pp. 169–180. [CrossRef]
87. Ly, L.T.; Knuplesch, D.; Rinderle-Ma, S.; Göser, K.; Reichert, M.; Dadam, P. SeaFlows Toolset—Compliance Verification Made Easy. In Proceedings of the CAiSE Forum 2010, Hammamet, Tunisia, 9–11 June 2010; Soffer, P., Proper, E., Eds.; CEUR Workshop Proceedings: Hammamet, Tunisia, 2010.
88. Ly, L.T.; Rinderle-Ma, S.; Dadam, P. Design and Verification of Instantiable Compliance Rule Graphs in Process-Aware Information Systems. In Proceedings of the 22nd International Conference on Advanced Information Systems Engineering, CAiSE 2010, Hammamet, Tunisia, 7–9 June 2010; Pernici, B., Ed.; Springer: Berlin/Heidelberg, Germany; Hammamet, Tunisia, 2010; pp. 9–23. [CrossRef]
89. Herrestad, H. Norms and Formalization. In Proceedings of the 3rd International Conference on Artificial Intelligence and Law, ICAIL 1991, Oxford, UK, 25–28 June 1991; ACM: Oxford, UK, 1991; pp. 175–184.
90. Elgammal, A.; Turetken, O.; van den Heuvel, W.J.; Papazoglou, M. Formalizing and appling compliance patterns for business process compliance. *Softw. Syst. Model.* **2016**, *15*, 119–146. [CrossRef]
91. González, L.; Delgado, A. Towards compliance requirements modeling and evaluation of E-government inter-organizational collaborative business processes. In Proceedings of the 54th Hawaii International Conference on System Sciences, HICSS 2021, Kauai, HI, USA, 5 January 2021; ScholarSpace: Kauai, HI, USA, 2021; pp. 1–10.
92. Ly, L.T.; Maggi, F.M.; Montali, M.; Rinderle, S.; van der Aalst, W. A Framework for the Systematic Comparison and Evaluation of Compliance Monitoring Approaches. In Proceedings of the 17th IEEE International Conference on Enterprise Distributed Object Computing Conference, EDOC 2013, Vancouver, BC, Canada, 9–13 September 2013; IEEE: Vancouver, BC, Canada, 2013; pp. 7–16.
93. Novotná, T.; Libal, T. An Evaluation of Methodologies for Legal Formalization. In *Explainable and Transparent AI and Multi-Agent Systems*; Calvaresi, D., Najjar, A., Winikoff, M., Främling, K., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 189–203.
94. Wang, W.; Chen, T.; Indulska, M.; Sadiq, S.; Weber, B. Business process and rule integration approaches: An empirical analysis of model understanding. *Inf. Syst.* **2022**, *104*, 101901. [CrossRef]
95. Fenech, S.; Pace, G.J.; Okika, J.C.; Ravn, A.P.; Schneider, G. On the Specification of Full Contracts. *Electron. Notes Theor. Comput. Sci.* **2009**, *253*, 39–55. [CrossRef]
96. Prisacariu, C.; Schneider, G. A Formal Language for Electronic Contracts. In Proceedings of the 9th IFIP WG 6.1 International Conference on Formal Methods for Open Object-Based Distributed Systems, FMOODS'07, Paphos, Cyprus, 6–8 June 2007; Bonsangue, M.M., Johnsen, E.B., Eds.; Springer: Paphos, Cyprus, 2007; pp. 174–189.
97. Hoare, C.A.R. *Communicating Sequential Processes*; Prentice Hall: Upper Saddle River, NJ, USA, 1985.
98. Allaire, M.; Governatori, G. On the Equivalence of Defeasible Deontic Logic and Temporal Defeasible Logic. In Proceedings of the 17th International Conference on Principles and Practice of Multi-Agent Systems, PRIMA 2014, Gold Coast, QLD, Australia, 1–5 December 2014; Dam, H., Pitt, J., Xu, Y., Governatori, G., Ito, T., Eds.; Springer International Publishing: Gold Coast, QLD, Australia, 2014; pp. 74–90. [CrossRef]
99. Breaux, T.D.; Antón, A. Analyzing Regulatory Rules for Privacy and Security Requirements. *IEEE Trans. Softw. Eng.* **2008**, *34*, 5–20.
100. Kiyavitskaya, N.; Zeni, N.; Breaux, T.D.; Antón, A.I.; Cordy, J.R.; Mich, L.; Mylopoulos, J. Extracting Rights and Obligations from Regulations: Toward a Tool-supported Process. In Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering, ASE '07, Atlanta, GA, USA, 5–9 November 2007; ACM: Atlanta, GA, USA, 2007; pp. 429–432.
101. Sapkota, K.; Aldea, A.; Younas, M.; Duce, D.; Bañares Alcántara, R. Extracting Meaningful Entities from Regulatory Text: Towards Automating Regulatory Compliance. In Proceedings of the International Workshop on Requirements Engineering and Law, RELAW 2012, Chicago, IL, USA, 25 September 2012; IEEE: Chicago, IL, USA, 2012; pp. 29–32. [CrossRef]
102. Wieringa, R.J.; Meyer, J.J.C. Applications of Deontic Logic in Computer Science: A Concise Overview. In Proceedings of the 1st International Workshop on Deontic Logic in Computer Science, DEON 1991, Amsterdam, The Netherlands, 11–13 December 1991; Meyer, J.J.C., Wieringa, R.J., Eds.; John Wiley & Sons, Inc.: Amsterdam, The Netherlands, 1993; pp. 15–43.
103. Ferraro, G.; Lam, H.P. NLP Techniques for Normative Mining. *J. Appl. Logics* **2021**, *8*, 941–974.
104. Ferraro, G.; Lam, H.P.; Colombo Tosatto, S.; Oliveri, F.; van Beest, N.; Governatori, G. Automatic Extraction of Legal Norms: Evaluation of Natural Language Processing Tools. In Proceedings of the 13th International Workshop on Juris-Informatics, JURISIN 2019, Kanagawa, Japan, 11–12 November 2019; Sakamoto, M., Okazaki, N., Mineshima, K., Satoh, K., Eds.; Springer International Publishing: Kanagawa, Japan, 2019; pp. 64–81. [CrossRef]

105. Halpern, J.Y.; Lakemeyer, G. Multi-agent Only Knowing. *J. Log. Comput.* **2001**, *11*, 41–70. [CrossRef]
106. Dinesh, N.; Joshi, A.; Lee, I.; Sokolsky, O. Permission to speak: A logic for access control and conformance. *J. Log. Algebr. Program.* **2011**, *80*, 50–74. [CrossRef]