ISSN: 2454-9940

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

E-Mail :
editor.ijasem@gmail.com
editor@ijasem.org

www.ijasem.org

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN2454-9940www.ijsem.org
Vol 5, Issuse.4 Oct2017

Applications of Deontic Logic in ComputerScience: A Concise Overview

Mohd Irshad , Gayatri , Mohd Afzal

Abstract
Actual and ideal system behavior are both considered within the scope of deontic reasoning. In this article, we provide a comprehensive overview of the many eighties-era uses of deontic logic in computer science, along with a methodology for categorizing such uses. A growing number of uses are emerging for deontic logic, which involves instructing a computer to forbid, allow, or oblige users to do certain actions. We analyze the scenarios where this is possible and the ones where it is acceptable.

Introduction

Ideal and real conduct may be reasoned about using deontic reasoning. Deontic logic is a kind of modal logic that includes operators for permission, obligation, and prohibition. It was created beginning in the 1950s by scholars including Von Wright [62, 64], Castan eda [12], and Alchourro'n [1]. The set of ideas developed by Hohfeld in 1913, which includes operators for duty, right, power, obligation, etc. [20], might be formalized to include more operators.

Normative law and normative legal reasoning have long been studied through the lens of deontic logic. Therefore, it is not surprising that the first interest in using deontic logic in computing sprang from the field of law. Many articles on the use of deontic logic in the optimization of legal automation may be found in the

Hosted every other year since 1987, this conference has recently published many papers on the topic of deontic logic's potential solutions to AI-related legal issues.

Deontic logic has been around for a while, but it's only lately been understood that it has applications outside legal analysis and legal automation. Any field where we wish to reason about both the ideal and real behavior of systems might benefit from the use of deontic logic. We will look at the design of fault-tolerant systems, the definition of security rules, the automation of contracts, and the specification of normative integrity requirements for databases as examples of practical uses of computer science.

First, we provide a short, approximately historical overview of the ways in which deontic logic has been used in computer science (section 2). In the third part, we adopt a more methodical stance and attempt to group all conceivable applications into a small number of broad categories. By doing so, we can establish some order in the prior study and find potential new areas of application. Finally, we consider some of the restrictions on deontic logic's applicability that arise from its unique character as a medium to dictate human conduct.

Asst. Professor
Department of CSE
mohdirshadislclg@gmail.com, gayatri12islclg@gmail.com, mdafzal9islclg@gmail.com,
ISL Engineering College.
International Airport Road, Bandlaguda, Chandrayangutta Hyderabad - 500005 Telangana, India.

1 A chronological survey of applications

1.1 Legal automation

When we talk about "legal automation," what we mean is the use of technology to facilitate activities traditionally associated with the law. All the way from EDI and information retrieval to providing legal counsel, this may cover a wide variety of tasks. In this paper, we focus on the latter category and investigate how deontic logic might be used to automate the provision of legal counsel. When it comes to using computers to provide legal advice, there are two schools of thought: the factual school of thought, which makes no difference between reality and ideality, and the deontic school of thought, which does make such a distinction. We next briefly address this distinction before turning to some concrete instances of deontic techniques.

1.1.1 The factual and deontic approaches to legal automation

In 1957, Layman Allen [3] noted that formal logic may be used to spot loopholes in laws and extrapolate reasonable outcomes from existing regulations. In turn, this may aid lawmakers in removing unnecessary ambiguities and streamlining the language of statutes. Allen used first-order logic without deontic operators to explain this idea in two publications published in the early 1980s [4, 5]. His method has been applied in genuine legislation [18] in the state of Tennessee.

In 1985, the logic programming group at Imperial College implemented a section of the British Na- tionality Act of 1981 in the computer language Prolog [54]. Legislation is seen as a series of definitions rather than a set of duties, permis- sions, and prohibitions given by authorities in this method of formalizing the law. So, a set of rules in Prolog formalizes the idea of British citizenship as stated by the 1981 U.K. statute. This is another case of formalizing legislation in accordance with the facts, like Allen's.

Other instances of the fact-based strategy include legal expert systems, which often fail to accurately portray the rationale behind the distinction between ideal and real-world contexts. Thorne McCarty's TAXMAN [38] system, for instance, models the legal issues underlying company reorganization under U.S. tax law. The basic idea behind the approach is to define the definitions of these notions and apply them to the realities of a reorganization to determine whether or not the reorganization may be regarded as a tax-free transaction.

The field of legal automation is rife with such instances of fact-based techniques. The depiction of law in computer systems is an area that Marek Sergot [53] examines in depth, including both factual and deontic methods. Although his focus in another survey work [52] is on the factual approach, he does address philosophical concerns with formalizing and automating legal advice regardless of methodology, including the deontic one. Below, in Section 4, we address a few of these concerns.

Andrew Jones [24] observed that the factual method is acceptable so long as the only goal is to ascertain how the legal definitions of concepts relate to the specific situation being investigated. Clarifying what the legislation really says or means may be aided by a structure designed along these lines (according to the interpretation used in formalizing the text of the law). However, the capacity to consistently articulate breaches of these concepts is a characteristic of deontic logic, but it is lacking in this method. Sergot [51] brought out the requirement for such logic in the depiction of law as logic programs in 1982. Jones provides an example of relatively straightforward normative assertions whose formalization necessitates confronting some of the most intractable difficulties in deontic logic. In

a nutshell, Jones demonstrates the following restrictions from the Imperial College library: where is a patron and is a loanable item:.

should return by the due date 1.

No disciplinary action will be taken against you if you return your book before the due date.

Third, disciplinary action will be taken anyone someone does not return by the specified date.

For the moment, assume the following is correct: If (4) fails to return by the specified date, we have an instance of what is known as Chisholm's dilemma, or the paradox of contrary-to-duty imperatives, in the field of deontic logic [13]. Paradoxically, a fair formalization of these phrases suggests that there is both a responsibility to discipline and an obligation not to discipline. To avoid this paradox, one must either reduce the case to a simple one (because the truth of sentence 4 renders the truth of sentence 2 trivial), or solve some of the most difficult issues in philosophical logic, such as the formalization of counterfactual conditionals. Some of the concerns involved are critically surveyed by Tomberlin [57], and in a recent study, Jones and Po rn [25] present a resolution of the contradiction using a modified understanding of duty. In his discussion, Meyer [43] provides a

potential answer based on dynamic reasoning. However, this problem is still being investigated and is far from being settled.

Jones and Sergot provide further analysis of reasons for the use of deontic logic in the design of normative system behavior [26].

1.1.2 Examples of deontic approaches

From the middle of the 1970s onwards, at Imperial College, a group headed by Ronald Stamper used a deontic method in a project called LEGOL. The LEGOL project aimed to create more realistic representations of reality via the use of conceptual modeling techniques during the creation of information systems inside businesses. For this purpose, the "traditional" approach to formal modeling based on denotational semantics was judged insufficient, thus a new method was devised that focuses on behaviors and social norms instead. LEGOL may be categorized as a legal automation project due to one of its use cases, which is the representation of law in computers. The project's output is a language for specifying models of information systems that is similar to relational algebra but includes operators for dealing with time [27]. One of the linguistic expansions includes operators for handling deontic ideas like right, obligation, privilege, and culpability [55]. However, nobody followed through on the plan to use deontic reasoning. The search for an adequate semantics for the modeling of business processes continues, but now at the University of Twente, where Stamper has relocated and started a new project.

Research that followed the TAXMAN project is another early example of using deontic logic for the computer representation of legal reasoning. One of the problems with the TAXMAN project's initial, factual approach was that it couldn't adequately depict the contrasts between ideal and real situations of the world. The computerization of corporation tax legislation still necessitates this distinction. For instance, the permissions and responsibilities incumbent on the business and its securityholders are the sole way to distinguish between certain types of stocks and bonds [40]. So, McCarty developed a form of dyadic deontic logic and reported on it in articles published in 1983 and 1986 [40, 41] to expand his method to include deontic reasoning in law. His deontic operators are embedded in a language that also includes constructions for describing activities. Language for Legal Discourse (LLD) [42] is a subset of a broader general-purpose language that also includes components for describing categories and subcategories, events, and timestamps. To illustrate the Lisp-like syntax of LLD, consider the following rule [41, page 323]: "Any business that holds cash has a duty to give cash to all its investors." The word "oblige" is an oxymoron.

(own ? Company X, Inc. (cash ?Y)) Publish a dividend (distribute-dividend?) company X)
Named variables X and Y are denoted by their respective letters, whereas anonymous variables are indicated by question marks.

The ESPLEX system, presented in a 1987 work by Biagioli et al. [10], is yet another example of the use of deontic logic to legal automation. The ESPLEX system is rule-based. under Italy's agricultural tenancy law, which dates back to 1982. In ESPLEX, for instance, you may use a Prolog-like syntax to provide a condition under which evicting a tenant is allowed.

> Allowed (cancellation, tenant, lease): condition (tenant farmer) > procedure (tenancy termination).

In order for the rule's conclusion to hold, the prefix cond specifies a necessary predicate, and the prefix proc specifies a necessary legal process stated elsewhere in the system. No reasoning behind Biagioli et almethodology .'s is provided.

In 1985, Layman Allen and Charles Saxon [7] demonstrated how to use Hohfeld's conceptual framework to develop a formal language for conducting in-depth, exact analyses of legal documents for the purpose of disambiguation. Ideas like as right, responsibility, privilege, authority, liability, and immunity are included here, as are variations on the more basic deontic concepts of permission, obligation, and prohibition. In this book, they demonstrate how their method may be applied to a specific set of rules—the Imperial College Library Regulations [6]—and demonstrate that there are, in fact, 2560 potential interpretations of these rules. They provide a mechanism for producing alternative interpretations (called MINT) that may aid those who issue rules (in government and commercial organizations) in finding ambiguity and clarifying the wording of the rules.

Next, we'll talk about how these systems are supposed to be put to work in practice. LEGOL was created for the express purpose of defining normative features in business models. McCarty's focus is on the real practice of lawful thinking [42]. Although the paper's [10] purpose for ESPLEX is not specified, it seems to be an expert system shell that may be used to construct expert systems in numerous areas of law. It is expected that these expert systems will be utilized to provide guidance on how to apply the law to specific situations. Finally, Allen's work is not focused on assisting with the application of law to real issues, but rather on assisting lawmakers in the planning and writing of texts that represent law in both public and private spheres. In Section 3, we'll talk in detail about the many contexts in which advice-giving systems might be useful.

1.2 Authorization mechanisms (Minsky and Lockman 1985)

One way in which actors are given the green light to act is via the process of authorization. In computer science, authorization procedures are used to safeguard sensitive resources, such as those found in operating systems and databases, as well as to preserve the confidentiality of personal information. Existing authorisation methods, as highlighted by N. Minsky and A. Lockman [45] with considerable force in 1985, are flawed due to the absence of the idea of an obligation. In the first place, the idea of having to do something is important in its own right, whether or not one has been given permission to do so. Locking data at the start of a database transaction, for instance, prevents other users from making changes to the data while the transaction is in progress. When the begin transaction event is executed, it is assumed that the transaction will be committed or rolled back within a reasonable amount of time.

or a reversed deal. This means that if a person acts in a specific way, he will be obligated to do further action.

Minsky and Lockman believe that granting rights with no conditions attached is generally insufficient as an authorisation method. The following illustrations are provided.

Permissions to conduct actions, such as reading a file or updating a database field, are often provided without imposing any obligations on the actor at the time of the action being performed. Nonetheless, such a duty often exists by implication. For instance, if I am allowed to borrow books from a library, I will be responsible for returning those books after I have finished reading them. This responsibility has to be spelled out in the same way that the privilege of borrowing books is defined.

A computer system's limitations may be temporarily disregarded, but doing so will result in the need to restore them. Assume,

for the sake of argument, that we want to authorize someone to assign workers to positions, with the proviso that critical positions would never be empty for more than five days. We therefore would want to be granted authorization to assign positions while also requiring that if an individual is released from a critical position, a replacement be found by the following weekend.

Let's pretend we give someone permission to do things in set 1, and then give that same individual permission to do things in set 2 on their own. There may be situations in which carrying out an activity from one set prohibits carrying out an action from the other set, meaning that these permissions may not be additive. If we can condition our permissions such that carrying out an activity from one set imposes a duty on the grantee to refrain from carrying out an action from another set, then we will have achieved our goal. one person's response to another's activity.

The language Minsky and Lockman suggest would allow for the "stringent" expression of permissions. As an example, the employee allocation authorization may be stated as

can	**where**	**and**
	requiring to do	**by**
	or else	

In cases when the employee's position is essential to the operation of the department, this indicates the authority to release them from duty; nevertheless, if this release is carried out, it will trigger a responsibility to fill the position by the following weekend. It is assumed that there is a system in place to ensure that promises are kept. The enforcement mechanism acts if the duty is broken.

Syntactic structures for dealing with layered commitments, deadlines, triggers, and negative actions are proposed by Minsky and Lockman (refraining from action). They provide no logical justification for these constructions, simply an informal semantics.

Specification of the System 2.3 (Khosla and Maibaum 1987)

In VDM, a system's behavior is explicitly defined with the use of preconditions and postconditions. Preconditions are used to define both the possible outcomes of an activity and the conditions under which that action may take place. Commonly, when describing database transactions, preconditions are used to guarantee that, if satisfied, a set of static limits on the database's allowable states will not be violated.

Khosla and Maibaum [29, 28] note that this calls for two separate preconditions to be applied, and they recommend keeping them that way. First, the postcondition of an action must specify the setting in which the action is performed before the result can be stated. If that were all that preconditions did, then the lack of one for an action would signify not that it could happen whenever it wanted to but rather that its result was unaffected by its context. In addition, you may utilize preconditions to define the circumstances under which an action is permissible. Due to the absence of a prior required and sufficient condition, this action may occur at any moment.

To distinguish between the two uses of preconditions, Khosla and Maibaum propose limiting the former to the statement of an action's outcome and the latter to the language of deontic logic. If the permission to do an action can be defined separately from the action's preconditions and postconditions, then it may be easier to specify the action's consequences. Second, keeping this difference allows for the specification of fault-tolerant systems, which are required whenever unwanted behavior, such as hardware failure, cannot be removed completely. Therefore, deontic logic may be used to outline the steps that need be taken to undo or at least lessen the effects of bad system behavior.

According to Khosla and Maibaum, Deontic Action Logic is an extension of modal action logic (DAL). A more colloquial description of DAL is that it "bans" or "allows" any and all possible system conditions. To begin, if the system is in a permissive state, any acts that lead to other permissive states are likewise permitted, and all actions that lead to additional forbidden states are prohibited. Furthermore, it is not specified whether or not actions that cause a system to enter an illegal state are permitted. If the system is in an unlawful condition, we could forbid all activities, permit just those that lead to legality, or selectively permit certain behaviors that do not get the system closer to legality.

In [29], Khosla and Maibaum detail a DAL-based telephone infrastructure. This specification makes a few assumptions, including:

Without specifying the context in which they will be carried out, the first three axioms determine the outcome of three actions. To indicate that the called phone (the callee) is busy in case (1), to indicate that the called phone (the callee) is ringing in case (2), and to ring its bell in response to the signal in instance (3), the called phone will sound a tone. In the last two axioms, we see what the call system should do when it determines that the called party is busy (4) or available (5). (5). (5). For this reason, the busy signal is sent to at the intervals defined in (4). After the connect operation, the exchange must inform the calling party () that the called party is ringing, and the calling party must simultaneously send a bell-ringing signal to the called party ().

DAL employs modal action logic and has operators like parallel composition, sequential composition, and choice to incorporate actions into more complex processes. In a similar vein of research, Jose' Fiadairo of INESC has been collaborating with Tom Maibaum [16] to include deontic specification into a generic specification language. Why this reduction to temporal logic is problematic is something we explore in our companion piece on deontic logic [44].

"2.3 The Use of Electronic Contracting" (Lee 1986)

It was noted in 1984 by Kimbrough, Lee, and Ness [30] that workplace papers often

have both informational and performative significance. For instance, the information included in an order placed by a client and sent to a supplier is useful since it includes the customer's name and address as well as product identifiers. It also has performative value since it comprises a query to the provider to deliver products, which the supplier ought to respond, and a commitment of the consumer to pay for the items when they are provided. A signature or other measures of authentication are often used to express this performative value.

These days, with the help of workplace information systems, computers may be used to store and even alter both instructive and functional documents. For instance, if the client and provider are linked by an EDI network, the order to restock the client's inventory may be transmitted to the provider at regular intervals, such as the month's end. Therefore, it is crucial for the advancement of office information systems to conduct a study of both the informational and the performative structure of these papers.

However, the performance features of the information system are not addressed by the conventional approaches of information construction. The performative structure of the data or of the manipulations of the data is not explicitly represented in data models, which instead utilize a subset of first-order logic to express the structure of the data. Therefore, conventional approaches need refinement in order to include performative considerations.

The general logic of the performative function of information systems should be based on speech act theory and might be some type of illocutionary logic [49]. However, some sorts of performative activities might be codified using less heavy tools. Deontic logic's potential as a representational framework is briefly discussed by Kimbrough et al.

framework for contract-related performance obligations. For example, key acts in contracting whose logic must be expressed include

Oblige	Some action not obligated becomes obligated
Waive	Some obligated action becomes not obliga

Lee figured out how to implement this in a contracting-monitoring system in 1988 [32]. Contracting challenges often revolve on the portrayal of processes and real-time in order to achieve deadlines. Lee employs Von Wright's [63] logic of change to represent processes in his logic, and Petri nets to provide them with a semantics. Time limits are represented using the logic of absolute time proposed by Rescher and Urquhart [47]. Deontic operators are introduced using a variation of Anderson's [8] reduction of conventional deontic logic to alethic logic. To make this multi-hued specification language actionable, a Prolog-like language is derived from it, and then a natural language interface is tacked on. Using it, contracts like these may be formally specified.

Jones agrees to pay \$500 to Smith by May 3, 1987. Following that, Smith agrees to deliver a washing machine to Jones within 30 days.

The system may them be questioned as follows:

?- at 5-may-1987 whatif nothing.

to which it responds with

Part Jones defaults at May 4, 1987,
because Jones failed to pay $500 to Smith by May 3, 1987.

The specification language is not provided with a formal semantics or inference mechanism. In addition, Sandra Dewitz [15], a Ph.D. student of Lee's, focuses on automating contracts using EDI networks.

In the same year (1988), Lee released another work ([31]) that examined administrative structures via the

lens of normative rules. He uses the same kind of language to specify the rules for issuing parking permits on a college campus, and then demonstrates how a rule-based system can be used to determine whether or not actors are authorized to carry out a given set of operations, and, if so, how this authorization derives from the rules.

2.3 Constraints on the Deontic Integrity (Wieringa, Meyer and Weigand 1989)'

Database integrity constraints are mathematical rules that the database's states and state transitions must satisfy in order to be properly defined. For instance,

It is a static restriction that a person's age in years cannot be less than zero. Salary freezes and no pre-employment terminations are also examples of dynamic limitations. In a 1989 study, we suggest that there is a significant split between what we term essential constraints and what we call deontic constraints. Knowing that each database stores information about a specific slice of the actual world helps to shed light on the discrepancy. Real-world necessary restrictions are formulae that can't be broken by the world as it really exists. One such limitation is that a person's age must be positive and cannot be negative. In other words, it is an analytic fact that follows logically from the meaning of the words used to describe the constraint, and as such, it cannot be broken in the actual world. For the same reason, the restriction that no one may be laid off before he is recruited is only a mathematical certainty that has no bearing on actual conduct. Given the present state of our language, these are analytic facts that do not impose any constraints on the states or behaviors of the actual world; nonetheless, they may be used to restrict the conceivable states and behaviors of the database. For the same reason that a record in a historical database of a fire event that was not preceded by a hiring event is not a realistic portrayal of reality, neither can an age field in the database include a negative value. Obviously, such a database is in error.

The set of required constraints for a database model may be

enlarged to include all conceivable states of the universe in which our interests lie, as well as all empirical facts that are not analytical truths. For instance, in all of the states of the globe we care about, the maximum age of a living human being, as measured in years, is 150. However, it is true in the sense that our experience has led us to believe it, and it is theoretically falsifiable by the existence of an universe in which it is not (without that being a result of a change in our use of our language). Nonetheless, for the sake of data modeling, we may consider this constraint as if it were a purely analytical fact, as it holds true in all possible world states that the database will ever reflect. As such, we may also use it as a restriction on the database's potential states, dismissing any condition in which a person is given an age greater than 150. Obviously, this is only possible if we leave plenty of room between ourselves and the point at which empirical truth could cease to be true. For instance, using the factual reality of the assertion that a person cannot have an age exceeding 100 as a restriction on the potential states of the database is too risky. It is reported that this limitation slowed down the whole database system of a major insurance firm in the Netherlands since one of their customers had reached the ripe old age of 101.

In [59], we make the case that many or perhaps most of the illustrative cases of database constraints

published in the literature are normative assertions that apply to the actual world and that may be broken in the real world, not essential truths in the sense stated above. In light of the aforementioned limitation that salaries must not be cut (a favorite example of many database researchers). When the world breaks such a restriction, it is not because we have redefined terms or because an empirical generalization has been disproved; rather, it is a breach of a standard in the actual world. Therefore, it is a limitation of the physical universe, rather than a set of truths.

about the external world. It is expected that deviations from this standard will be represented in any database containing information about this region. Most importantly, it must be able to portray this as an aberration rather than just another real truth. Since the distinction between a pay increase and a decrease is a reality of the world in which we are engaged, its absence would be a disservice to the numerous possible applications we have in mind. As a result, deontic logic emerges as the best choice for defining such restrictions in a database. 1 We use an Anderson-like reduction of deontic logic to dynamic logic, as detailed in our companion study of deontic logic [44]. This approach resembles the reduction of DAL to action logic proposed by Khosla and Maibaum. Therefore, we classify every nation as either restricted or liberated. In contrast to DAL, our reasoning prohibits any activity that would lead to an illegal state of the world and approves any action that would lead to a legal state of the world. Furthermore, we describe the cause of the violation in the violation predicate, which allows us to identify the most effective means of resolving the problem and to provide more helpful error messages. Also, the relationship between the three modal operators (permission, prohibition, and obligation) is defined with extensive use of the idea of action negation. Other distinctions include the semantic structure we describe for specifications, and the usage of propositional negation (to guarantee deterministic processes). Another research of ours [61] investigates the issue of deontic constraint inheritance in a taxonomic structure. Actors, initiative, and action negation are the focus of recent studies [60].

We have a specification language that allows us to declare limitations like as . Every behavior has a necessary condition. When this condition holds, we state that a violation flag has been raised because has not returned in a timely manner. According to subsection (7), everytime the violation flag: is raised, on must pay a fine of $. According to the rule, the violation flag is reduced when the book is returned.

We should probably refer to deontic integrity requirements as "real-world constraints" instead, as they impose limitations on the physical world rather than on a database. Since they only serve to limit the data in the database and not the world at large, necessary database constraints should be referred to by that name alone. However, the phrase "integrity constraint" is so deeply rooted in the lexicon of the database community that we continue to use it, modifying it with the adverbs "deontic" and "necessary" to indicate whether we are referring to real-world restrictions or database constraints.

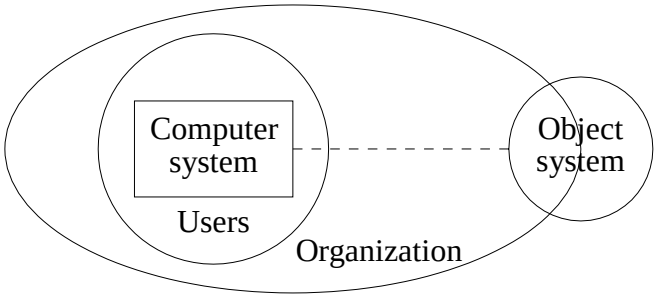

Figure 1: The structure of computer applications.

Privacy in Data Storage (2.3) (Glasgow, MacEwen and Panangaden 1989)

Database security rules are analyzed using deontic logic by Glasgow, MacEwen, and Panangaden [17]. They use a hybrid of epistemic logic (with both positive and negative introspection) and deontic logic (where the "user is authorized to know that") For, we provide the following axioms, where denotes "user knows that."

2 A systematic view of applications in computer science

Looking at the structure of any computer program, as illustrated in figure 1, provides a basic systematic framework for understanding the aforementioned applications of deontic logic to computer science as well as additional conceivable applications. It's safe to say that every computer system ever built has the capacity to store and alter data. Databases, expert systems, knowledge-based systems, decision-

support tools, operating systems, etc., all fall under the umbrella of "computer system." Data that is of relevance to us may be found in the types of computer programs that symbolize a slice of the actual world. The object system of the application is the representation of the world. Users provide data into the system, make demands of it, and consume its output (answers). A company's users and its IT infrastructure together form a subsystem. It's possible that certain components of the object system may be internal, while others will be external to the business. (It might be the computer system itself or comprise components of it.)

By examining the domain whose behavior is given in deontic logic, we may use this framework to categorize the uses of deontic logic in computer science. All of the described actions are both real and ideal examples of such actions. As a result, we may categorize apps as follows.

Six, computers that can recover from errors without crashing.

7. Common patterns of action among users.

8. Typical actions inside a company.

For (a), we need a definition of the policy.

(a) Typical organizational conduct (e.g. contracting).

Nine, the object system's typical actions.

In particular: (a) legal enunciation.

The articulation of legally-minded thought (b).

Normative rule specification as deontic integrity constraints (c).

Aside from the aforementioned contexts, (d) several more uses. The potential applicability to scheduling issues is further upon below.

More in detail, the list of possible applications is as follows.

1. **Fault-tolerant computer systems**. There is no such thing as a completely bulletproof computer system, and there are times when we need to outline procedures for dealing with hardware failure or other deviations from the norm. Fault-tolerant computer systems may exhibit less-than-ideal behavior that is nonetheless contextually relevant. You may look at this use of deontic logic as the definition of managing computer-generated exceptions. Khosla and Maibaum's method is one such implementation. Some of the scenarios described by Minsky and Lockman, in which restrictions on a computer's behavior are momentarily relaxed, also come under this category.

2. **Normative user behavior**. Users act in a wide variety of ways, many of which are undesirable. They could make typos, enter information that conflicts with what's already in the system, ignore the system's requests for information, raise queries they're not meant to, etc. While some of this activity may be picked up by the computer system, other instances may not, therefore it's important to differentiate between the two. that which is expected of the user and what the user could really do. This would enable us to more easily express desirable user behavior, as well as what should happen in the event that the user does not act as expected. The statement of how to deal with user-generated mistakes may be seen as an example of deontic logic in action here. None of the preceding apps even touch on this topic. Third, deontic logic's applications to businesses may be broken down into two categories: (1) those that deal with the conduct of workers, and (2) those that deal with the conduct of the business itself.

For (a), we need a definition of the policy. Deontic logic, which considers the organization as a whole, may be used to provide guidelines for the conduct of its various parts, such as individuals or departments. This is done in the form of organizational policies, which are intended to serve as a set of rules for conduct. Deontic logic is useful because policymakers often wonder what will happen if their policies aren't implemented. For instance, deontic logic may be used to eliminate any ambiguity in the policies and to investigate the effects of altering their specifications. Policies are not laws since they are issued by private entities, but their application is comparable to that of the legal area.

Deontic logic is used as a subset of policy specification for defining security policies. Even yet, this does not imply that deontic reasoning is utilized to code safe computers. An attribute of trustworthy computer systems is that they forbid any deviation from the established security guidelines. On the other hand, deontic logic may be used to create the security rules, investigate the effects of those policies, and even provide proof that a given piece of software adheres to the policy by disallowing any actions that would be considered malicious. Examples from Minsky and Lockman, as well as Glasgow et alstudy, .'s fit into this category of uses.

That which is considered to be the "b" norm in a certain organization. By using deontic reasoning, we may also provide guidelines for how an organization should act in its external context. Consider Lee's view of the contracting process. Organizational conduct should be lawful, hence this may be seen as a subset of the legal application of deontic reasoning.

Four, in deontic logic, describing how the object system ought to act. The object system is a portion of the physical world that must have its details encoded in a computer. This category includes anything from a library to an elevator system, and serves as a catch-all for the applications we haven't yet discussed. In light of the aforementioned evaluation, we have discovered the following programs.

Law is defined in deontic logic (a). Without the use of computers, deontic logic has a wide range of applications in the legal system. Figure 1 depicts the object system in this kind of application, which comprises of humans and a computer network. of rules and regulations governing their actions. The principles of certain deontic logic are used to codify and manipulate facts and laws in order to arrive at conclusions. A legal advising system like this might be utilized in a number of contexts, such as a teaching tool for law students, during the drafting of legislation, or while implementing new laws. The second kind of application is shown by Allen's work and the planned usage of ESPLEX. There are examples of each of these applications in Sergot [53]. Section 4 will address a problematic fourth use case: providing legally binding advice via the system. There, we'll also talk about issues with the naive perspective of law application previously presented, which affect even the more limited advice-giving usage of deontic logic in legal automation.

Second, using deontic logic as a model for legal reasoning. McCarty's method of legal automation differs greatly from that of his predecessors since it attempts to mimic the way in which human attorneys and judges genuinely reason. Here, the formalized object system is not a set of rules and authorities but rather a mental operation being carried out by a small group of experts. Since

this mental process is concerned with truths and standards, deontic reasoning might prove beneficial. On the other hand, the computer system's representation of the object system in this instance cannot be mistaken for a representation of law. To the contrary, it reflects the evolution of legal professionals' conceptualizations of the law. Deontic logic is not being used here to dictate conduct in the object system (thought process activity), but rather as a vehicle for expressing empirical hypotheses regarding the nature of this behavior (thinking about normative systems).

Constraints on deontic integrity must be specified (c). An easy example of formalizing rules that pertain to humans in the object system is the declaration of deontic integrity constraints. Both our own application and the examples provided by Minsky and Lockman fit into this category. (d) Diverse usages As a potential use not previously stated, it may also be used to timetable conflicts. In such situations, jobs and resources must be assigned with a set of restrictions. Many databases, for instance, need to execute processes at regular intervals (once a week, for example), in accordance with certain restrictions that impose a sequential order on the execution of the processes and, in certain cases, a deadline on the actual time that a process is performed. Because computers are imperfect devices, it is important to define the behavior expected in the event that a constraint is broken. Interesting applications include the assignment of tasks to machines, the parking of planes at airports, the seating of passengers on planes, the response of police to incidents, and the distribution of other types of widgets and fidgets, all subject to normative scheduling constraints that may be disregarded in practice.

Finally, a comment on the above categorization is warranted. In every conceivable context, Specification of both normative and behavioural standards is central to deontic logic. However, the stated uses are not limited to those that need a computer-executable implementation of the deontic logic specification. However, although this recommendation holds especially water in the context of fault-tolerant systems and legal automation, it is still technically unnecessary. Even if these systems aren't computerized, a deontic characterization of their behavior might be valuable. Therefore, we separate the use of deontic logic for describing system behavior from its use in computer programming. These two situations coincide if the system is specified in an interpreted form of deontic logic. However, in most situations, this is not the case, and distinguishing between them is helpful.

Discussion: directing human behavior by computer

The majority of computer science's uses for deontic logic include the field's ability to prescribe human action. Applications in which norms relevant to human conduct are stated in deontic logic include the specification of user behavior norms, organization policies, organizational behavior, legislation, deontic integrity restrictions, and even certain scheduling difficulties. The unique scenario arises if the specification is implemented in a computer, at which point the machine may actively derive human rights, duties, and restrictions. There has never been a case like this one before in which deontic reasoning has been put to use. However, allowing computers to take charge of human affairs is not a novel concept in computer science, even though the implementations haven't been very problematic up to this point. According to Sergot [53], even a payroll computer uses the law to compute tax deductions and thereby establishes an individual's legal entitlement. The usage of traffic lights is only one example that shows how far back the concept of computers prescribing human behavior goes. However, if we get to the point where we can implement deontic logic requirements in computers, the scope and complexity of this usage of machines expands enormously. Therefore, the feasibility and acceptability of using computers to

guide human affairs should be included in any comprehensive examination of the uses of deontic logic in computer science. The potential and legal basis for this action are discussed. The possibility to direct human affairs by computer

We will focus our conversation on the controversial topic of using AI to make decisions for people, which has sparked heated debate in the areas of AI and the law. Nonetheless, the topic is equally applicable to the enforcement of corporate regulations on employees or to any of the other uses of deontic logic in computer science where the application of standards for human conduct is automated. A common misconception amongst computer scientists is that the process of applying law to facts is conceptually identical to how a computer follows instructions to process data. This is the perspective we used while compiling the following comprehensive list of ways in which deontic logic has been applied to computer science. However, the reality is more complicated, and in order to evaluate the likelihood of utilizing computers to influence human behavior, we must simply to the discrepancies between how humans and computers might interpret standards (such as rules) and apply them to data. For a human judge to properly apply the law to a set of circumstances, both the facts and the law must be read in light of one another. 2 Law in The Netherlands may be found in a variety of government-issued statutes, judicial precedents, generally recognized norms of society, and even a few international treaties. In each given situation, only a subset of this will apply, and even then, the applicable parts will be found in the form of broad generalizations that will need to be narrowed down to fit the specifics. A selection is chosen from the many possible sources of law, and those sources are construed such that they apply. Decisions and analyses are made in the context of the evidence under consideration. On the other hand, as information analysts are well aware, everything is interconnected in the end, and a decision must be made based on a theoretically endless collection of potentially relevant data in every given situation. If a police officer sees a car parked illegally on a sidewalk, he or she will record the vehicle's license plate number and the date and time of the observation, but not the color of the car, the temperature outside, or the width of the sidewalk, unless any of those details seem pertinent. As an added bonus, the legal definitions of any words used in the observation will be included. For instance, the word "vehicle" has a specific legal meaning that is laid down. To make the applicable law applicable, it is necessary to choose some of the many possible relevant facts and interpret those facts in light of the law. The applicable laws provide the lens through which all decisions and interpretations must be made. Let's compare this to what occurs when we teach a computer to take a digitized version of the law and apply it to a digitized version of the facts. Starting with the British Nationality Act [54], corporate tax law [38], or latent damage law [11], a specific topic of law is chosen to examine. This decision is made not by a court considering circumstances that may constitute a breach of law, but rather by a legal professional (or a knowledge engineer) considering the practicality of expressing this area of the law in a computer. We'll discuss the factors that go into this determination, such as the level of technical difficulty and the degree to which it stands alone from the rest of the law. Whatever criteria are used, they are applied before any particular set of facts is chosen to represent a certain instance.

The next step is to create a digital version of the statutes and caselaw that have already been codified. This translates into an exercise in interpretation, but not in the light of facts to be tested (as they aren't ready), but in the light of one or more people's understanding. For

instance, the Latent Damage Law system represents the thinking of P.N. Capper [11], while the British Nationality Act was coded in Prolog in accordance with the knowledge of F. Sadri [54].

Third, when the system is applied to facts, these facts have to be presented to the system in some form suitable for storage and manipulation. The individual carrying this out must use their own judgment in this regard. The system has to be given with the information on the basis

2

This is discussed in great detail in a textbook used as an introduction to the philosophy of law at various Dutch colleges [2]. Sergot [53, pp. 18-3 1] provides a helpful description of debate, often confined to the philosophy of law, about whether judges do nothing more than apply laws to facts, and situates this debate within the framework of the computer representation of law. For a more in-depth analysis, check out Susskind's dissertation [56].

Given words known to the system, and because these terms are computer representations of previously interpreted terms stated in the law, this is quite different from the actual interpretation of events in light of the law that happens in practice.

At last, the computerized application of law to the computerized analysis of facts. In the end, all a computer does is apply instructions to data, but this process is better understood when characterized at a higher level as the manipulation of data in accordance with particular principles. The rules in question could be those of a specific branch of first-order logic, like declarative Prolog, or of a branch of first-order logic that is less well-known, like procedural Prolog (which includes mechanisms like cut that are not well-understood), a rule-based mechanism, or a variant of deontic logic. Whatever the method, it's important to note that it's not meant to, and in fact doesn't, reflect how judges and attorneys really think about the law.

It's important to note that the prior decisions and interpretations (of facts and laws, of manipulating rules) that have played a part are now invisible. This is why many people believe that a computer's application of law to a computer's representation of facts is more objective and less prone to bias than a judge's application of law to facts. Of course, this is not the case; rather, the employment of a computer only renders the decisions and interpretations invisible [33]. We should not state that the computer lacks bias or is objective, but rather that its partiality and subjectivity are hardwired into it.

Simply said, the path that leads to a computer applying (a computer representation of) law to (a computer representation of) facts is quite different from the one that leads to a human judge applying law to facts. When a computer model of the law is applied to a computer model of the facts, just as when a judge applies the law to facts, choices are made and interpretations are made. However, the choices made by the computer model of the law are independent of the choices made by the computer model of the facts, and both are hidden once the model of the law has been applied to the model of the facts. In the absence of a human translator, a computer model of the law may be applied to a model of the facts, creating the appearance of objectivity.

Now that the differences between computerized legal processing and computerized legal representation by machines have been established, it remains unclear how the latter may contribute to the former. The following circumstances may be distinguished that, in one way or

another, sidestep the issues caused by the dissimilarities between a human judge's application of law to facts and a computer representation of law applied to a computer representation of facts.

In most cases, the chosen domain of law has little bearing on other fields of law that may be represented in a computer. This helps keep the system's size reasonable and prevents conflicts with words stated in other portions of the legislation. This need is met by all of the aforementioned examples of domains of law (corporate tax law, agricultural tenancies, and British citizenship) chosen for computer representation.

Using the computerized version of the law takes nothing in the way of common sense. As a result, we can sidestep the challenge of communicating with AI, which is often considered to be among the field's most intractable issues.

computerized model of human common sense. For a further discussion of this topic, you may like to see Thorne McCarty [39].

The chosen field of law has a clear meaning that is accepted by other lawyers. By doing so, we may sidestep the issue of a mysterious interpreter injecting bias into the system under the appearance of objective technology. This is a necessary condition, as stated explicitly by Sergot [53].

The chosen domain of law is immutable, or at least immutable in the sense that it may be altered via a continuous jurisprudential process. This saves us the trouble of teaching the computer to engage in social learning as if it were a human being. Another challenging issue in AI is learning, and social learning is much more difficult. On a more practical level, picking stable areas of law facilitates system maintenance by reducing the need to manually update a computer representation of law. 3

Conclusions

Following a chronological overview, we categorized (though not exhaustively) the various uses of deontic logic in computer science as follows: the specification in deontic logic of fault-tolerant systems; the desired behavior of users; the policies of businesses (including security policies); the actions of organizations; and so on (i.e. contracting),

scheduling under normative restrictions, legal reasoning, and limits on normative integrity.

In all of these scenarios, the discrepancy between the specified and actual behavior is important, and we have the option to program to the specification. We concluded by noting that many uses of deontic logic in computer science involve the robotic production of normative statements about individuals. We pointed out that the question of who has the authority for these automated decisions and who is responsible for them should be resolved before such systems are actually used, and we identified routine problems that satisfy a number of conditions as potential applications of this kind of automation.

Exciting in the use of deontic logic in computer technology is the combination of difficult philosophical questions of law with practical concerns of social morality. This motivates the need for further multidisciplinary studies of these topics and how they relate to one another.

References

Source: [1] C.E. Alchourro'n and E. Bulygin. Systems of Norms. 1971, Springer.

Algra, N.E., and van Duyvendijk, K. Legal primer: introductory lessons in law and legal studies from the textbook An Introduction to Law and Legal Studies. 4e druk. Samsom Tjeenk Willink, H.D.

As cited in [3] L.E. Allen. Symbolic logic is a precise instrument for working with legal texts. The year 1957 saw Yale Law Journal's 66th volume cover the pages 833-879.

Language, law, and logic: Plain drafting for the digital era [4], by L.E. Allen. Computing Law and Policy, edited by B. Niblett, pp. 75–100. In 1980, Cambridge University Press published this.

Allen, L.E. Aiming for a standardized vocabulary to better define the framework of legal discussion. The article may be found in volume 2 of A.A. Martino's Deontic Logic, Computational Linguistics, and Legal Information Systems (pages 349-407). 1982, North-Holland. Presented here are revised and edited versions of talks presented during the international conference "Logic, Informatics, Law," held in Florence, Italy, in April

1981.

Allen, L.E., and Saxon, C.S., [6]. A-Hohfeld is a language for constructing expert systems that provide interpretation help by means of a strong structural representation of knowledge from the legal area. The present volume.

Allen, L.E., and Saxon, C.S. A codified and updated version of Hohfeld's essential legal ideas is used to analyze the logical structure of legal norms. Automated Analysis of Legal Texts, edited by A.A. Martino and F.S. Natali, pages 385–450. It was published in North-Holland in the year 1986. This volume contains revised and abridged versions of talks presented at the Second International Conference on "Logic, Informatics, Law," held in Florence, Italy, in September 1985.

Anderson, R.A. [8]. normative system analysis in formal logic. Pages 147–213 in The Logic of Decision and Action, edited by N. Rescher. Dated 1967 by the University of Pittsburgh Press.

Theresa Bench-Capon and Marcel Sergot [9]. Intent on representing open texture in the law in terms of a set of rules. Published on pages 39–60 in C. Walter (ed.), Computer Power and Legal Language. 1988, Quorum Books.

A. Tiscornia, D. Biagioli, and C. Biagioli. The ESPLEX Paradigm is a rule and conceptual based model for expressing legislation. Pages 240–251 of The Proceedings of the First International Conference on Artificial Intelligence and Law. In ACM (May 1987).

According to [11] P.N. Capper and R.E. Susskind. Expert Methods in Latent Damage Law. Dated 1988 by Butterworths.

As stated by H.-N. Castan eda [12]. Putting one's thoughts into action. What Institutions Stand on Philosophically. In the year 1975, Reidel published it.

A quote by R.M. Chisholm (see footnote #13). Contrary to deontic reasoning and moral imperatives. Analyses, 24(33):36 (1963).

For more reading, see [14] C. Ciampi (editor). First volume on the intersection of AI and LIS. 1982, North-Holland. Presented here are revised and edited versions of talks presented during the international conference "Logic, Informatics, Law," held in Florence, Italy, in April 1981.

[15]

Dr. S. D. Dewitz. Using information technology as a legal mediator in contracts based on a performative network. Collaborative Work, Social Communications and Information Systems, edited by R.K. Stamper, P. Kerola, R. Lee, and K. Lyytinen, pages 271-293. A 1991 publication from North-Holland.

Fiadeiro, José, and Maibaum, Timothy (16). Legal norms should give way to temporal logic.

It will appear in the Journal of Logic and Computing.

John Glasgow, Gordon MacEwen, and Peter Panangaden [17]. Databases are secured by permission-based security. Database Security II: Status and Prospects, edited by C.E. Landwehr, pages 197-205. 1989, North-Holland. Database Security: Outcomes from the IFIP WG 11.3 Workshop, Kingston, Ontario, Canada, October 1988.

Referenced in [18] G.B. Gray. Tennesseans have had some practice in enacting statutes in a standard format thanks to their state legislature's history. Pages 467-493 of Computer Power and Legal Reasoning, edited by C. Walter. 1984, West Publishing Co.

H.J. van den Herik [19] Can a Machine Have Morality? The year 1991 for Gouda Quint B.V.

W.N. Hohfeld, page 20. Concepts fundamental to the law and the process of deciding cases.

Journal of the Yale Law School 23 (1913): 16–59.

[21] Proceedings of the First Annual Conference on the Intersection of AI and the Legal System. May 1987, Association for Computing Machinery.

[22] Proceedings from the 2016 AI and Law Conference, held in London. From June 25-28, 1989, ACM held its annual conference.

[23] Proceedings from the 2018 AI&CJ International Conference. Date: June 25-28, 1991, Association for Computing Machinery.

AJI Jones, 24. Moral reasoning and the codification of legal precedent. Published in Ratio Juris 3 (1990):237-244.

To paraphrase, [25] A.J.I. Jones and I. Po rn. There are three levels of ideality: the ideal, the sub-ideal, and deontic. Synthese, 1985, 65:275–289.

As cited by A.J.I. Jones and M. Sergot [26]. This book focuses on the function of deontic logic in defining normative structures.

Jones, Patrick Mason, and Robert Stamper [27]. Complex rule specification in LEGOL 2.0, a relational specification language. The Journal of Computer-Based Information Systems 4 (1979): 157–169.

The idea originated with S. Khosla [28]. A Deontological Method for Defining Systems. Thesis for a doctoral degree in computer science, Imperial College London.

Credit goes to S. Khosla and T.S.E. Maibaum [29]. State-based system description and prescription. Pages 243-294 in Temporal Logic in Specification, edited by B. Banieqbal, H. Barringer, and A.

Pnueli. This was published by Springer in 1987. The 398th issue of Lecture Notes in Computer Science.

S.O. Kimbrough, R.M. Lee, and D. Ness [30]. The PIE framework's first component is made up of systems that are both functional and empathetic. 1984: pages 141-148 in L. Maggi, J.L. King, and K.L. Kraenens's Proceedings of the Fifth Conference on Information Systems.

Reference: [31] R.M. Lee, Bureaucracies as Deontological Systems. In 1988, ACM published volume 6 of its Transactions on Office Information Systems, which had articles 87-108.

[32] R.M. Lee. The logical framework for digital agreements. Systematic Approaches to Decision Making, 4(1-2):27-44 (1988).

Source: [33] M. Lupoi. Computerized decision making as a legitimate legal authority. North-Holland, 1982, "Artificial Intelligence and Legal Information Systems," edited by C. Ciampi. Presented here are revised and edited versions of talks presented during the international conference "Logic, Informatics, Law," held in Florence, Italy, in April 1981.

Citation: [34] A.A. Martino (editor). North-Holland, 1982. Deontic Logic: Computational Linguistics and Legal Information Systems. Presented here are revised and edited versions of talks presented during the international conference "Logic, Informatics, Law," held in Florence, Italy, in April 1981.

The modifying editor was A.A. Martino (35). Proceedings of the Third International Conference on "Logic, Informatics, Law," Florence, Italy, 1989.

Martino, A.A., and Natali, F.S., editors. Machine-Readable Legal Text Analysis. It's 1986 in North Holland. This volume contains revised and abridged versions of talks presented at the Second International Conference on "Logic, Informatics, Law," held in Florence, Italy, in September 1985.

Referenced in: L.V. Mazel [37].

Fully automated, as an example of practical use.

Trias Automatica, edited by E.M.H. Hirsch Ballin and J.A. Kamphuis, pages 89–94. For the 1985 edition, see Kluwer.

McCarty, L.T. Analysis of the Artificial Intelligence and Legal Reasoning Experiment TAXMAN. During the month of March in 1977, the Harvard Law Review was published at Volume 90:837-893.

L.T. McCarty, cited in footnote [39]. The features that should be included in a digital legal advisor. Presented at the 1980 National Conference on Artificial Intelligence, Stanford, August. Presented at the First Annual National Conference on Artificial Intelligence, pages 298–300, Proceedings.

Permissions and Duties, by L.T. McCarty. From 1983's Eighth International Joint Conference on Artificial Intelligence (IJCAI), held in Karlsruhe, West Germany, the paper was published on pages 287–294. Kaufmann.

According to L.T. McCarty (reference number 41). An unofficial introduction to permissions and duties. Automated Analysis of Legal Texts, edited by A.A. Martino and F.S. Natali, pages 307. It was published in North-Holland in the year 1986. papers presented at the 2nd International Conference on "Logic, Informatics, Law," edited and revised for publication. Sept. 1985, Florence, Italy.

[42]

Essential elements of a language for legal discussion (I) by L.T. McCarty. Pages 180–189 in The Proceedings of the Second International Conference on Artificial Intelligence and Law. June 25–28, 1989, Association for Computing Machinery.

In reference to: [43] J.-J.Ch. Meyer. An easy way out of the 'deepest' conundrum in deontic reasoning.

1987's "Logic and Analysis: New Series" (Vol. 30:pp. 81–90).

According to Meyer, J.-J.Ch., and Wieringa, R.J. A synopsis of deontic logic. The present volume.

A.D. Lockman and N.H. Minsky [45]. Adding responsibilities to rights is one way to guarantee honesty. Pages 92-102, 1985 IEEE International Conference on Software Engineering.

As cited in [46] M.A. Nieuwenhuis. TESSEC is an Advisory System for the Uniform Commercial Code. 1989 Ph.D. diss., University of Twente.

To cite: [47] N. Rescher and A. Urquhart. Time-Based Reasoning. 1971, Springer.

J. Searle [48]. Expressions of the Verbal Mind. An Exposition on the Theory of Language. The year 1969 saw the publication of this work by Cambridge University Press.

A reference to the work of J. Searle and D. Vanderveken [49] is in order. The Ilocutionary Logic Foundations. Oxford University Press, 1985.

J.R. Searle, page 50. Classification of nonverbal behaviors. Pages 1–29 in Expression and Meaning. Oxford University Press, 1979.

Law as a Logic Program: Future Prospects, by M. Sergot [51]. Logic Programming, edited by K.L. Clark and S.-A. Ta rnlund, pages 33–42, Academic

Press, 1982.
Referencing M. Sergot's Representing Legislation as Logic Programs [52]. Machine Intelligence 11, edited by John E. Hayes, David Michie, and John Richards, pages 209-260. Dated 1988 by Clarendon Press.
The representation of law in software: A review and comparison, by M. Sergot [53]. Published in T.J.M. Bench-(ed. Capon's Legal Knowledge Management Systems. In 1990, Academic Press published.
Reference: Sergot, F. Sadri, R.A. Kowalski, F. Kriwaczek, P. Hammond, and H.T. Cory (54). Applying logic to the British Nationality Act. 1986 saw a focus on pages 370-386 for the ACM's Communications journal.
[55] R. Stamper. A computerized model of the rules of law; sometimes known as LEGOL. Com[puter Science and Law, edited by B. Niblett, pages 45-71. In 1980, Cambridge University Press published this.
According to R.E. Susskind (reference 56). A Legal Analysis of Expert Systems. A publication of Oxford University Press.
[57]
It was J.E. Tomberlin. There are conditional duties and imperatives that go counter to what one ought to do. Nouˆs, 16:357–375, 1981.
[58]
To wit: R.J. Wieringa. Here are three ways in which conceptual models are used in the development and management of IT systems. The Case of E.D. Falkenberg Reference: Information System Concepts: A Deep Dive, edited by P. Lindgreen, pp 31-51. 1989, North-Holland.

R.J. Wieringa; J.-J. Ch. Meyer; H. Weigand [59]. Defining bounds on the kinetic and deontological integrity. Reference: Data and Knowledge Engineering, 4:157–189 (1989).
Wieringa, R.J., and Meyer, J.-J.Ch. Normative system definition including actors, actions, and initiative. Report IR-257 from the Department of Mathematics and Computer Science at the University of Amsterdam's Vrije Universiteit, published in October 1991. Postmarked for publishing.
According to Wieringa, Weigand, Ch. Meyer, and Dignum [61], R.J. Integrity restrictions, both dynamic and ethical, that are inherited. 1991, 3:393-428 in the Annals of Mathematics and Artificial Intelligence.
(G.H. von Wright, Deontic Logic, p. Mind, 60:1–15, 1951.
From the works of G.H. von Wright (ref. And then what? North-Acta Holland's Philosophica Fennica Volume 18 from 1965.
According to G.H. von Wright (referenced in [64]). Theory of Action and Deontic Logic: A Dissertation. Volume 21 of Acta Philosophica Fennica, published in 1968 by North-Holla